Documentation for OpenKM 7.0 CE

Table of Contents

Table of Contents 2
OpenKM 7.0 CE 33
Hardware and software requirements 35
Requirements 35
Checking the Java version 35
For Windows users 36
Hardware configuration 36
Browser support 37
Operating systems 37
For deployment: Windows Server 2003, Windows Server 2008, Windows Server 2012 38
Small questionnaire 38
Third-party software requirements 40
Installation 41
Installing on Ubuntu and Debian 43
Preliminaries 43
Installing OpenKM 44
Post-installation checklist 44
Troubleshooting 45
Installing on Red Hat and CentOS 46
Preliminaries 46
Installing OpenKM 47
Enable remote access 48
Post-installation checklist 48
Troubleshooting 48
Installing on Windows 49
Preliminaries 49
Checking Java version 49
Install Java 49
Installing database 49
Check InnoDB is the default MySQL engine 50
Create database 50
Installing Tomcat 50
Configuring Tomcat as a service 50
Check the service 55
From terminal 55
From the Windows service panel 55
From openkmw tool 55
Configuring environment variables 56
Installing OpenKM 58
Start application 59
Check openkm.properties parameters 59
Access from other computers 60
Other standard additional setup configurations 60
Using the installer 61
Supported databases 62
Other options 63
Using Docker 64
Configuration 65
Docker Compose 66
Installing on a pristine Tomcat 69
Preliminaries 69

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

Installing Tomcat 69
Copy libraries and configuration files 69
Modify configuration files 69
Installing OpenKM 70
Install third-party software 70
Start the application 70
Check the openkm.properties parameters 70
Database 72
Configure the database 72
Configuring MySQL and MariaDB 73
Preliminaries 73
Change default engine 73
Change default charset 73
Database creation 74
Create a database and user 74
Configure your openkm.properties 74
MySQL Timezone 74
Configure Tomcat data sources 75
MySQL: 75
MariaDB: 75
Configure application login 75
Run application 75
Additional information 76
Remove and create MySQL service in Windows. 76
Migrate from Windows to Linux 76
MySQL tuning 77
Sample one 77
Sample two 78
Change MySQL data directory 80
Ubuntu 80
CentOS and RHEL 81
Windows 82
MySQL 8 troubleshooting 83
Could not create a connection to the database server 83
Strong Password Encryption for Authentication 83
Old JDBC driver 85
The server time zone value is unrecognized. 86
Problem connecting to your database 86
Configuring Oracle 88
Database creation 88
Create tablespace 88
Create a database and user 88
Configure your openkm.properties 88
Configure Tomcat data sources 88
Configure application login 89
Run application 89
Configuring PostgreSQL 91
Database creation 91
Create a database and user 91
Configure your openkm.properties 91
Configure Tomcat datasources 91
Configure application login 92
Enable native configuration parameter 92
Run application 92
Additional information 92
PostgreSQL 9.6 and above 92
Configuring SQL Server 93

Database creation 93
Create a database and user 93
Configure your openkm.properties 94
Configure Tomcat data sources 95
Configure application login 95
Run application 95
Configure integrated security 96
Troubleshooting errors 96
Can't connect to SQL Server 96
The connection was lost after some time. 97
Unable to log in to OpenKM 97
Unable to connect via TCP/IP to localhost SQL, port 1433 when using an instance in the JDBC connection URL 98
Deadlock issues 98
SSL connection error 98
The driver could not establish a secure connection to SQL Server by using Secure Sockets Layer (SSL) encryption 98
SQL Server alter compatibility level 99
Configuring Hikari pool 100
Enable debugging 100
Backup and restore 101
Linux detailed steps 102
Backup 102
Restore 102
Windows detailed steps 103
Backup 103
Restore 103
Backup scripts 104
Additional information 104
Linux backup and restore with BorgBackup 105
Installation 105
Sample script 105
Backup to a remote computer with SSH 105
This is an improved SSH version 106
Backup to mapped network 106
Configure crontab 107
Recovery 108
Linux backup and restore with the rdiff-backup tool 109
Debian & Ubuntu 109
CentOS & RedHat 109
Preliminaries 109
Sample script 109
Increase the backup period 112
Do remote backup 112
Do a remote backup to SMB or CIFS 113
Do PostgreSQL backup 113
Configure crontab 113
Restoring backup 114
Additional information 114
Linux Sample Backup with Duplicity 116
Installation 116
Command line basics 116
Preliminaries 117
Sample script 117
Increase the backup period 120
Perform the remote backup 120
Perform the remote backup to SMB or CIFS 120
Do the PostgreSQL backup 121
Do the remote backup to Amazon S3 121
Configure crontab 122

Troubleshooting deleting old backups 122
Restoring backup 122
Additional information 123
Linux backup and restore with rsync 124
Preliminaries 124
Sample script 124
Do backup with rotation 126
Do remote backup 127
Do incremental backup with backup-dir 127
Do incremental backup with link-dest 128
Do remote backup to SMB or CIFS 128
Do PostgreSQL backup 128
Configure crontab 129
Rsync and FTP 129
Restoring backup 130
Additional information 130
Linux backup with LFTP 131
Preliminaries 131
Sample script 131
Configure crontab 133
Instance replication 134
Different implementations 135
Sample Linux implementation 135
How to move to another server 137
Linux 137
Preliminaries 137
Stop the OpenKM service 137
Backup the database 137
Restore the database 137
Copy binary files 138
Configure the service on the new server 138
Install third-party tools 138
Start the OpenKM service 139
Additional steps 139
Final checks 139
Setup 140
Antivirus configuration 141
Linux 141
Debian / Ubuntu 141
Red Hat / CentOS 141
Windows 142
Clamd Service Setup 143
Freshclam Virus Signature Auto Update 144
Configuring Apache HTTP Reverse-Proxy 146
Internal IP vs external IP 146
Debian and Ubuntu 147
Check the configuration 149
Red Hat and CentOS 149
Check the configuration 150
Error Permission denied: proxy: AJP: attempt to connect 150
Windows 150
Register Apache service 151
Apache SSL configuration 151
Troubleshooting 152
Additional information 153
Configuring Apache HTTPS Reverse-Proxy 154
Internal IP vs external IP 154

Debian and Ubuntu 155
Redirect all HTTP connections to HTTPS 157
Check the configuration 157
Using Let's Encrypt 157
Red Hat and CentOS 158
Check the configuration 159
Error Permission denied: proxy: AJP: attempt to connect 159
Windows 159
Troubleshooting 159
Additional information 160
Configuring Nginx HTTP Reverse-Proxy 162
Internal IP vs external IP 162
Debian and Ubuntu 163
Check the configuration 164
Red Hat and CentOS 164
Check the configuration 165
Configuring Lucene Analyzer 167
Configure an Analyzer 168
Configuring OCR engine 169
Tesseract 169
Installation 169
Check installation 170
Configuration 170
Additional information 171
Dictionary 172
Configuration 172
OCR rotate configuration 172
Configuration 172
Additional information: 172
Configuring OpenSSL 174
Create OpenSSL configuration file 174
Create a Certification Authority (CA) 175
Create Apache Certificate 176
Apache configuration 176
Certificate revocation 177
Add CRL to Apache 178
Configuring mail 179
Change mail configuration 179
Gmail example 179
Office 365 179
Email server with old SSL configuration 180
Additional information 180
Configuring Tomcat memory utilization 181
OpenJDK 1.8 181
Configuring Tomcat ports 182
Change Tomcat ports 182
Configuring Tomcat session timeout 184
Configuring Tomcat timezone 185
Configuring openkm.xml 186
Enable external openkm.xml 186
Memory sample 186
Database authentication sample 187
LDAP Authentication sample 187
Multiple providers sample 188

Configuring search highlighter 190
Configuring Tomcat to work behind an Internet proxy 191
Linux 191
Windows 192
Step one 192
Step two 193
Change repository home 195
Understanding the contents of repository folders 195
Move the entire repository location 195
Sample 1 196
Windows 196
Linux 196
Move repository folders individually 196
Sample 197
Windows 197
Linux 197
Troubleshooting 197
The Windows mapped network disk is not working 197
Change default role names 198
Change the default roles 198
Edit the openkm.properties file 198
Update the default roles to new names in the database 198
Enable extensions 199
LDAP configuration 200
OpenLDAP 200
Active Directory 200
Uncategorised examples shared by users 200
Tools 200
OpenLDAP example with URL base at login 201
LDAP Structure 201
Step 1 - configuration parameters 202
Step 2 - configure login 203
Another option to configure login 204
Active Directory example with referral enabled 206
LDAP Structure 206
Step 1 - configuration parameters 207
Step 2 - configure login 208
Another option to configure login 209
Active Directory example showing login based on filtering users by roles 211
LDAP structure 211
Step 1 - configuration parameters 212
Step 2 - configure login 214
Another option for configuring login 214
Active Directory basic configuration 217
LDAP structure 217
Step 1 - configuration parameters 217
Step 2 - configure login 219
Another option to configure login 220
Active Directory advanced configuration 221
LDAP structure 221
Step 1 - configuration parameters 221
Step 2 - configure login 223
Another option to configure login 224
Active Directory mixed configuration 225

LDAP structure 225
Step 1 - configuration parameters 225
Step 1 - configure login 227
Active Directory mixed configuration with LDAP synced 229
LDAP structure 229
Step 1 - configuration parameters 229
Step 1 - configure login 231
LDAP best practices for filtering users and roles 233
Example based on a single LDAP group 233
Example based on ROLE_USER and ROLE_ADMIN groups 233
ClearOS LDAP configuration 235
LDAP Structure 235
Step 1 - configuration parameters 235
Step 2 - create a Certificate 237
Step 3 - configure login 238
Another option to configure login 239
LDAP troubleshooting 241
How to perform a nested group membership filter in Active Directory 241
No UserDetailsService registered error 241
javax.naming.PartialResultException error 241
Can't connect to LDAP server 242
Slow login or not able to login 242
Force all users to be members of ROLE_USER. 243
org.springframework.beans.factory.parsing.BeanDefinitionParsingException 244
LDAP: error code 8 - BindSimple: Transport encryption required 245
LDAP error: Caused by java.security.cert.CertificateException: No subject alternative DNS name matching 246
Log Configuration 248
Logback-based Log Configuration 248
Third-party application execution 248
Enable log authentication 248
LibreOffice configuration 249
Linux 249
Debian & Ubuntu 249
Windows 249
Security configuration 251
Differences between simple and recursive 252
Extending security 253
Configuration for security changes which take a long time 253
SSO Configuration 255
Active Directory 255
Linux Machine 256
Kerberos configuration file 257
Configuration files 257
Configure LDAP in OpenKM 260
Client browsers: 260
Troubleshooting 261
Enable log for SSO authentication 261
Check Kerberos authentication from Linux command line 261
Listing keys 261
Configuring Two Factor Authentication 264
Compliance 264
Prerequisite 264

OpenKM configuration 264
Scan QR code 265
Google Authenticator use 265
Cache configuration 266
Configuring the anonymous user 267
Prometheus configuration 268
Setup validation 268
Troubleshooting 270
Server configuration 271
The packet for the query is too large (MySQL) 271
Not able to connect from another computer on the network 271
Error downloading document as PDF 271
IMAP SSL certificate error 271
The OpenKM server has no direct Internet connection. 271
Access denied when installing OpenKM as a service on Windows 7 272
Problems with character encodings 272
PermGen errors 272
LDAP Size limit exceeded 272
Behind the IIS proxy, you get error 404 or 404.15 273
IIS gets IOException: Internal Server Error while uploading a file or HTTP 413 Request Entity Too Large Error 275
Solution one 275
Solution two 275
WARN net.sf.ehcache.hibernate.strategy.AbstractReadWriteEhcacheAccessStrategy 276
Client configuration 277
ZIP filenames with wrong charset 277
Microsoft Office Add-in disappears after Office restart 278
Best practices for reporting issues 279
Getting the stack trace 279
Linux 279
Windows 279
Sample of stack trace capture 280
Administration guide 281
Configuration parameters 284
Edit a parameter 284
Delete a parameter 284
Backup configuration parameters 284
New configuration parameter 285
Check Configuration 285
Recommended configuration parameters 286
Preview parameters 289
Linux 289
Windows 290
LDAP configuration parameters 292
Sample query 292
LDAP queries 292
Parameters 293
Sample configuration 295
User interface configuration parameters 298
Notification configuration parameters 301
Substitutions 303

Document version numeration configuration 304
Version numeration schemas 304
Migration between versions 305
Password validation configuration 306
Sample configuration 307
Switch the application to read-only mode 308
Security configuration parameters 309
Extensions configuration parameters 313
ExtraTab extension 313
External tab document extension 313
External tab folder extension 313
External tab mail extension 313
External tab record extension 314
Performance configuration parameters 315
Text extractor queue 315
Download files and folders as ZIP file 316
Antivirus checker 317
Re-indexing the whole Lucene repository 318
Execution timeout 319
Activity log actions 320
Dashboard activity 320
Lucene 321
OpenOffice & LibreOffice configuration parameters 323
CSV configuration parameters 324
Antivirus configuration parameters 325
Other configuration parameters 327
Experimental configuration parameters 333
Two-factor authentication configuration parameters 336
MIME types 337
Add new MIME type 337
Edit MIME Type 338
Delete MIME type 338
Backup MIME types 338
New MIME types 338
Statistics 339
Check the status of the Pending task queue 339
Check the status of the Text extraction queue 339
Check the status of the Antivirus check queue 340
Metadata 341
Edit metadata 342
Metadata XML definition 343
Basic XML skeleton 343
Metadata group definition 343
Basic example 344
Using file system DTD 345
How metadata is stored into OpenKM 345
Metadata fields 348
Metadata Checkbox field 350
Checkbox 350

Metadata Iframe field 352
Metadata Select field 354
Basic select example 356
Allow multiple choices 356
Based on SQL query 357
Based on className 357
Suggested options 357
Hierarchical relationship 358
Hierarchical relationship with SQL 359
Metadata Separator field 361
Metadata Suggestbox field 362
Based on filterQuery and valueQuery 363
Based on className 364
Metadata text field 365
Metadata Textarea field 367
Metadata Validator element 369
Basic example 370
Regular expression 370
Metadata Option element 372
Basic select with selected option 372
Hierarchical relationship 373
Metadata Input field 374
Metadata VerticalPanel field 377
Metadata HorizontalPanel field 378
Metadata Security element 379
Basic grant example 379
Basic revoke example 380
Metadata internationalization 381
Label internationalization 381
Translations 381
Metadata definition 382
Manage users and roles 383
Manage users 384
Create a user 385
Edit a user 385
Delete a user 385
Change the user profile 386
Export the users 386
Manage user mail accounts 387
Create a new mail account 389
Edit a mail account 389
Delete a mail account 389
Browse emails on the mail server 389
Manage mail account filters 389
Create a mail account filter 391
Edit a mail account filter 391
Delete a mail account filter 392
Create a mail account rule 392
Edit a mail account rule 392
Delete a mail account rule 392

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

Manage roles 393
Create a role 393
Edit a role 394
Delete a role 394
Export the roles 394
Logged users 395
Real-time user activity log 395
Message queue 396
Send a message 397
Edit a message 397
Delete a message 397
Delete all messages in the queue 397
Refresh the message queue 397
Profiles 398
Create a new profile 398
Edit a profile 398
Delete a profile 398
Clone a profile 399
Profile General tab 400
General section 400
Misc section 400
Pagination section 401
Section General 402
Profile Components tab 404
Stacks section 404
Dashboard section 404
Tab section 404
Tab document section 405
Tab folder section 406
Tab mail section 406
Tab record section 407
Tab search section 407
Tab basic section 408
Tab advanced section 409
Profile Menu tab 411
Menu section 411
Menu bookmark section 411
Menu tool section 412
Menu help section 412
Menu file section 412
Menu edit section 413
Profile Notification tab 415
Events 415
Common 415
Document 415
Folder 416
Mail 417
Record 418
Profile Other tab 420
Toolbar section 420
File browser section 421
Relation types 423

Types 423
Bidirectional 423
Parent - child 423
Many-to-many 424
Create a relation 424
Edit a relation 424
Delete a relation 424
CSS 425
Create a new CSS 425
Edit a CSS 425
Delete a CSS 425
Clone a CSS 425
Download a CSS 426
Samples 426
Classical UI samples (frontend) 426
Large font sample 426
Medium font sample 426
Default font sample 427
New UI samples (KCenter) 427
Large font sample 427
Medium font sample 427
Arabic sample 427
Reports 429
Create a new report 429
Edit a report 429
Delete a report 430
Show report parameters 430
Execute a report 430
Reports pack 431
Template reports 431
Authentication 431
Documents 431
Folders 434
Mails 435
Others 436
Activity log 437
Auth actions 437
Document actions 438
Folder actions 439
Mail actions 440
Record actions 441
Property group actions 442
Node actions 443
Repository actions 443
Admin actions 443
Misc actions 448
Automation 449
Preliminary concepts. 449
Understanding Validators and Actions 451
Creating a rule 452
Editing a rule 452
Edit general parameters 452
Edit action and validation definitions 452
Deleting a rule 453
Built-in Automation Validator plugins 454

Built-in Automation Action plugins 458
Crontab 462
Create a new crontab job 464
Edit a crontab job 464
Execute a crontab job 464
Delete a crontab job 464
Built-in crontab tasks 465
Languages 467
Create a new language 467
Edit a language 467
Delete a language 467
Edit language translations 468
Export language translations 468
Import language translations 468
Language translations 469
How to create a new translation 472
A sample of a new language based on Oromo from Ethiopia 472
Utilities 473
Admin permissions 475
Change user access level 475
Cache stats 476
Clean all cache 477
Refresh cache stats 477
Enable cache stats audit trail 477
Disable cache stats audit trail 477
Check email 479
Check email 479
Check text extraction 480
Check text extraction 480
Clean conversion cache 481
Clean conversion cache 481
Create support zip 483
Database import 484
Import 484
Database query 486
Execute JDBC query 486
Execute Hibernate query 487
Allowed sentences 487
Hibernate stats 488
Information provided by Hibernate stats 488
Enable Hibernate stats 488
Clear Hibernate stats 489
Disable Hibernate stats 489
LDAP Search 490
List indexes 492
Search indexes 493
Enable show terms 494
Disable show terms 494
LogCat 495

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

Purge all logs 495
Download file as ZIP 495
Browse log file 495
Plugins 497
Profiling stats 500
Enable profiling stats 501
Disable profiling stats 501
Clear profiling stats 501
Refresh profiling stats 501
List of detailed calls 501
Purge data 502
Purge 502
Filter 502
Purge trash 503
Purge 503
Rebuild indexes 504
Rebuild 504
Repository checker 505
Check repository 505
Repository export 506
Export 506
Understanding output format 506
If metadata and history are not checked 506
If only metadata is checked 507
If only history is checked 508
If metadata and history are checked 508
Repository import 509
Import 510
Scripting 511
Execute a script as part of application starting process 511
Execute a script as part of application stopping process 511
Scripting in Automation 511
System properties 513
User guide 515
Menu bar 516
Quick search box 516
Toolbar 516
Desktop 517
Navigation 517
Browser 517
Properties 517
Search 518
Dashboard 520
Administration 521
Status 521
Quick start 523
Logging in 523
Create a taxonomy 523
Add documents 525
Edit documents 526
Search documents 528
Essential fundamentals 530

Basic concepts 530
Understanding taxonomy, keywords, categories and metadata groups. 530
Taxonomy 531
Keywords 531
Categories 532
Metadata 532
Plan your data organization 533
Good practice 534
Working with the menu bar 535
File 535
Edit 538
Tools 540
Bookmark 540
Templates 541
Help 541
Working with the toolbar 542
Working with the desktop 545
Navigation 545
Browser 548
Pagination 552
Properties 553
Tabs on nodes of type document 554
Tabs on nodes of type folder 556
Tabs on nodes of type mail 558
Tabs on nodes of type record 560
Shortcuts 562
Working with search 564
Search panels 564
Search panel 564
Operators and wildcards 565
Default search 565
Advanced search 566
Control panel 568
Results 569
Working with the dashboard 572
News 573
Activity log extension 574
Activity log 574
Digital Signature viewer extension 575
Verify a signed document 575
Extra tab workspace extension 577
Configuration 577
Sample 577
External tab document extension 579
Configuration 579
Sample 579
Configuration 579
Usage 580
External tab folder extension 583
Configuration 583
Sample 583
Configuration 583
Usage 584
External tab mail extension 586

Configuration 586
Sample 586
Configuration 586
Usage 587
External tab record extension 590
Configuration 590
Sample 590
Configuration 590
Usage 591
Forum extension 594
Create a new topic 594
Create a reply 594
Update a post 594
Delete a post 594
HTML editor extension 595
Features 595
Edit HTML 596
Editor panel 596
Preview panel 597
Using OpenKM from external applications 598
Microsoft Office Add-in 598
Scanner Client 598
Microsoft Office Add-in 599
Configuring the add-in 599
Editing documents from Microsoft Office 599
Importing emails from Microsoft Outlook 600
Add new translation 600
Known issues 601
Scanner client 602
Requirements 602
Configuring 602
Compression CCITT Group4 603
Setting destination folder 603
Setting scanner configuration 603
Scanning 603
Convert scanned images to PDF 603
Uploading 604
Add images to an existing PDF file 604
Remove pages from existing PDF file 605
Update guide 606
Changelog 607
OpenKM 7.0 Community 607
v7.0.0 607
Tomcat update 608
Quick method 608
Linux 608
Preliminaries 608
Tomcat replacement 608
Windows 610
Preliminaries 610
Tomcat replacement 611
Changes from version 6.3 614
Changes 614

Deprecated features from version 6.3 614
Development 616
OpenKM professional portable development environment 616
Javadoc 616
Personal development environment 618
Requirements 618
How to get the personal development environment 618
How to install 618
Troubleshooting 619
Importing the OVA File into a Non–VirtualBox Virtual Machine 619
Configuring IDE from a clean Eclipse 620
Requirements 620
Software: 620
Hardware: 620
Download Eclipse 620
Configure Java 620
Configure Tomcat server 622
Maven support for Eclipse 628
Subversion support for Eclipse 628
Configuring Maven in your OS 630
Maven tips 630
Memory problems 630
Configure Eclipse to use local Maven 631
Eclipse tips 632
Debug Tomcat 632
Debug remote Tomcat server 632
Eclipse configuration 632
Eclipse known issues and troubleshooting 635
Maven raises error 206: the filename or extension is too long 635
Development tips 636
Tomcat application deployment 636
Sample deployment script 636
Profiling application 638
Edit Tomcat startup scripts 638
Authentication 638
Monitoring Remote JVM Over SSH 639
HPROF 640
Application version compatibility table 641
Creating a JAR file with Eclipse 642
Images 642
Sample JAR file 645
Creating plugins 646
Register a new plugin 647
Creating your own Antivirus Analyzer plugin 648
Method description 648
Example of the ClamAV antivirus analyzer implementation 649
Creating your own Automation Action 650
Methods description 651
Understanding the env variable 652
Understanding the params variable 652
Description of available ParamType values: 653
Description of available Src values: 653

Detecting recursion 654
Example 654
Creating your own Automation Validation 657
Method descriptions 658
Understanding the env variable 659
Understanding the params variable 659
Description of available ParamType values: 660
Description of available Src values: 660
Example 661
Creating your own Crontab plugin 664
Method description 664
Example of the cron implementation 665
Understanding the crontab job execution context 666
Creating your own Autocomplete Form plugin 667
Method description 667
Example of the default form values implementation 668
Metadata definition 668
Creating your own Default Form Values 670
Method description 670
Example of the default form values implementation 671
Metadata definition 671
Creating your own Document Converter 672
Example of a Document Converter implementation 673
Creating your own Form Validator plugin 675
Single field validation 675
Methods description 675
Example of the Form validator implementation 676
Full form validation 677
Methods description 678
Example of the Form validator implementation 679
Creating your own NodeProperties plugin (customizing search result objects) 682
Method description 683
Example of NodeProperties implementation 683
Creating your own Option Select Values plugin 685
Method descriptions 685
Example of the Option Select Values implementation 686
Creating your own Principal Adapter 687
Method descriptions 688
Example of the Principal Adapter implementation 690
Creating your own security access manager 693
Method descriptions 694
Example of a security access manager implementation 694
Creating your own Suggestion Analyzer 697
Method descriptions 697
Example of a Suggestion Analyzer 698
Creating your own SuggestBox plugin 700
Method descriptions 700
Example implementation 701
Creating your own Text Extractor 703
Method descriptions 703
Example of the Text Extractor implementation 704
Creating your own Version Number Adapter 705

Method descriptions 706
Example of the Version number adapter implementation 707
Creating templates 708
Creating LibreOffice templates 709
Preliminaries 709
Creating the template 709
Date format 710
Dynamic tables 711
Configuring the sample 711
Creating a PDF template 713
Preliminaries 713
Creating the template 714
Date format 715
Configuring sample 715
Creating text templates 716
Preliminaries 716
Creating the template 716
Date format 717
Sample configuration 717
Creating your own login module 718
PrincipalAdapter sample based on properties 720
Creating your own parameters 724
Sample 724
Creating your own password validator 725
Sample 726
Creating your own reports 728
Introduction to reports based on SQL 729
Create a datasource 729
Testing SQL sample report 730
A closer look at the JRXML file 733
Query section 733
Fields section 734
Remote image section 734
Using fields in report 734
Introduction to reports based on Java 735
Testing the Java sample report 735
Understanding the code sample 737
Using the report 738
Reports with parameters 740
Understanding filtering 740
Params XML file 740
Using fields in report 740
Java sample 741
SQL sample 743
Params form fields 745
Basic XML skeleton 745
Basic example 745
Using a file system DTD 746
Params fields 747
Params Validator element 749
Basic example 750
Regular expression 750

Params Input field 752
Params Suggestbox field 754
Based on the table, filterQuery, and valueQuery 755
Based on filterQuery and valueQuery (without table attribute) 756
Based on className 757
Params Checkbox field 758
Checkbox 758
Params TextArea field 759
Params Select field 760
Basic select example 761
Allow multiple choice 761
Hierarchical relationship 761
Based on SQL query 762
Based on className 763
Suggested options 763
Params Separator Field 765
Params Text field 766
Database stuff 767
Web services 768
RESTful 769
Sample usage 769
Folder samples 771
Document samples 771
Search samples 772
Security samples 772
Metadata samples 773
Notes samples 774
Conversion samples 774
Swagger 775
How to access Swagger 775
Playing with Swagger 777
RESTClient sample 782
Login 782
Get roles 783
PHP Samples 785
Download sdk4php 785
Preview sample 785
Login 785
Document samples 786
createDocument 786
deleteDocument 787
getDocumentProperties 788
getContent 789
getDocumentChildren 790
moveDocument 791
copyDocument 792
extendedDocumentCopy 793
checkout 795
cancelCheckout 796
forceCancelCheckout 797
isCheckedOut 798
checkin 799
getVersionHistory 800

Folder sample 800
createFolder 800
getFolderProperties 801
deleteFolder 802
renameFolder 803
moveFolder 804
getFolderChildren 805
copyFolder 806
extendedFolderCopy 807
getContentInfo 808
purgeFolder 809
createMissingFolders 810
createFolderFromTemplate 811
PropertyGroup samples 813
addPropertyGroup 813
removePropertyGroup 814
getPropertyGroups 814
getAllPropertyGroups 815
getPropertyGroupForm 816
getPropertyGroupFormByNode 817
setPropertyGroupProperties 818
hasPropertyGroup 820
getRegisteredPropertyGroupDefinition 821
getPropertyGroupSuggestions 822
registerPropertyGroupDefinition 823
getPropertyGroupProperties 824
getPropertyGroupPropertiesByVersion 825
Search samples 827
find 827
findPaginated 828
Integrate preview with other web applications 830
Troubleshooting 833
Samples 834
Scripting samples 835
Script - Basic samples 837
Update repository statistics 837
Create some folders and set a property group 837
Show the number of documents, folders, and the size for a given path 837
Get path by UUID 837
Script - Compact documents with size equal to 0 839
Script - Convert TIF to PDF 840
Script - Count all child subfolders 841
Script - Exporter 842
Script - Export repository folder to server as a ZIP file 843
Script - Folder depth 844
Images 845
Execution 845
Results 845
Script - Import data from CSV file 847
Image 849
Register the metadata group 849
Repository view 849
Execution 850
Script results 851
Imported metadata 851
Script - Purge all folders into a parent 852
Script - Purge all users' trash 853

Script - Put documents from a folder and its descendants into the pending text extraction queue 854
Script - Read PDF form fields 855
Script - Recursive character renaming 856
Script - Recursive repository traversal 857
Script - Recursive security change 858
Script - Security change at a low level 860
Images 863
Initial security status 863
UUID of the folder 864
Execution 865
Log 866
Rebuild Lucene indexes 867
Security changes applied 868
Script samples for automation 870
Script for automation - Autonumeric 871
Register metadata group 872
Script for automation - Set unique document name based on metadata 873
Register an automation rule 875
Crontab samples 876
Crontab sample - Basic file system document importer 877
Register a new plugin 878
Images 878
Crontab task 878
Files on the application server 879
Result 879
Crontab sample - Document importer with metadata values 881
Register a new plugin 886
Images 886
Register metadata definition 886
Crontab task 887
Files on the application server 888
Result 888
Mail notification 888
Files imported into the repository 889
Crontab sample - Generate an HTML report 891
Register a new plugin 892
Crontab task - XML importer 893
Register a new plugin 898
Images 898
Crontab task 898
Register metadata definition 899
Logfile folder 900
Output folder 900
Result 900
Variables by automation event 902
Available environment variables 902
Document 905
Create Document 905
Update Document 906
Delete Document 906
Rename Document 907
Purge Document 907
Move Document 908
Restore Version Document 908
Download Document from UI 909

Download Document from UI for preview 910
Folder 910
Create Folder 910
Delete Folder 911
Purge Folder 912
Move Folder 912
Rename Folder 913
Export Folder as ZIP 913
Export Folder as JAR 914
Mail 914
Create mail 914
Move Mail 915
Download mail from UI 916
Import mail attachments 916
Delete Mail 916
Purge Mail 917
Record 917
Create record 917
Delete record 918
Purge record 919
Move Record 919
Property Group (metadata) 920
Add property group 920
Set property group 920
Remove property group 921
Auth 922
Create user 922
User login 922
User logout 922
Note 922
Create Note 923
Update Note 923
Delete Note 924
General 924
Text extraction 924
Export nodes to zip 925
Vault of stuff 926
Convert Exchange Legacy DN 927
Download a document with a direct link 929
Import documents with a bash script using REST 930
Creating users from scripts 931
API description 932
OKMActivity 934
Methods 934
findByFilter 934
countByFilter 934
getActions 935
OKMAuth 937
Basics 937
Methods 937
login 937
login 938
grantUser 939
revokeUser 940
getGrantedUsers 940
grantRole 941
revokeRole 942
getGrantedRoles 942
getUsers 943
getRoles 944

getUsersByRole 944
getRolesByUser 945
changeSecurity 945
overwriteSecurity 946
createUser 947
deleteUser 947
updateUser 948
createRole 949
deleteRole 949
updateRole 950
assignRole 950
removeRole 951
getProfiles 951
getUserProfile 952
setUserProfile 952
logout 953
getUser 954
setUserPermissions 954
setRolePermissions 955
getUserTenants 956
setUserTenant 956
isPasswordExpired 957
resetPassword 957
OKMBookmark 959
Basics 959
Methods 959
add 959
get 960
remove 960
rename 961
getAll 961
OKMDashboard 963
Basics 963
Methods 963
getUserCheckedOutDocuments 963
getUserLastModifiedDocuments 964
getUserLockedDocuments 965
getUserLockedRecords 966
getUserLockedFolders 967
getUserLockedMails 969
getUserSubscribedDocuments 970
getUserSubscribedFolders 971
getUserSubscribedRecords 972
getUserLastCreatedDocuments 973
getUserLastCreatedDocumentsNotes 974
getUserLastCreatedFolders 975
getUserLastCreatedFoldersNotes 977
getUserLastCreatedRecords 978
getUserLastCreatedRecordsNotes 979
getUserLastDownloadedDocuments 980
getUserLastImportedMails 981
getUserLastCreatedMailsNotes 982
getUserLastImportedMailAttachments 983
getUserDocumentsSize 985
getUserSearches 985
find 986
getLastWeekTopDownloadedDocuments 987
getLastMonthTopDownloadedDocuments 988
getLastWeekTopModifiedDocuments 989
getLastMonthTopModifiedDocuments 990
getLastModifiedDocuments 991
getLastCreatedDocuments 992
getLastCheckedOutDocuments 993
getLastCreatedFolders 994
getLastCreatedRecords 995

visitNode 996
OKMDocument 999
Basics 999
Methods 999
create 999
createSimple 1000
delete 1001
getProperties 1001
getContent 1002
getContentByVersion 1003
getExtractedText 1004
getChildren 1004
rename 1005
setProperties 1006
setLanguage 1006
setTitle 1007
checkout 1008
cancelCheckout 1008
forceCancelCheckout 1009
isCheckedOut 1009
checkin 1010
checkin 1011
lock 1012
unlock 1012
forceUnlock 1013
isLocked 1014
getLockInfo 1014
purge 1015
move 1016
copy 1016
copy 1017
extendedCopy 1018
getVersionHistorySize 1019
isValid 1019
getPath 1020
getDetectedLanguages 1020
createFromTemplate 1021
updateFromTemplate 1022
setAnnotations 1023
getAnnotations 1024
getCheckedOut 1024
stamp 1025
stampText 1026
stampImage 1027
setDescription 1028
setNodeClass 1028
setDispositionStage 1029
createWizard 1030
getNumberOfPages 1030
getPageAsImage 1031
isConvertibleToPDF 1031
setIndexable 1032
mergePdf 1033
OKMFolder 1034
Basics 1034
Methods 1034
create 1034
createSimple 1035
getProperties 1035
delete 1036
purge 1036
rename 1037
move 1038
copy 1038
copy 1039

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

extendedCopy 1040
extendedCopy 1041
getChildren 1042
getContentInfo 1042
isValid 1043
getPath 1044
setStyle 1044
createMissingFolders 1045
addNodeClassChildren 1045
setDescription 1046
OKMImport 1047
Basics 1047
Methods 1047
createDocument 1047
createFolder 1048
createRecord 1048
createMail 1049
importFolder 1050
OKMMail 1052
Basics 1052
Methods 1052
getProperties 1052
createAttachment 1053
deleteAttachment 1053
getAttachments 1054
delete 1054
purge 1055
rename 1056
move 1056
copy 1057
copy 1057
extendedCopy 1058
extendedCopy 1059
getChildren 1060
isValid 1061
getPath 1061
sendMail 1062
setTitle 1062
sendMailWithAttachments 1063
importEml 1064
importMsg 1065
setNodeClass 1066
sendMail 1066
sendMailWithAttachments 1067
setDescription 1068
getContent 1069
createWizard 1070
getExtractedText 1071
forwardEmail 1072
OKMNode 1073
Basics 1073
Methods 1073
getNodeByUuid 1073
getVersionHistory 1074
restoreVersion 1074
renameVersion 1075
purgeVersionHistory 1075
getChildrenPaginated 1076
getChildrenByCategoryPaginated 1077
getBreadcrumb 1079
subscribe 1079
unsubscribe 1080
importZip 1080
unzip 1081

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

exportZip 1082
getNodesFromUuids 1082
evaluateZipDownload 1083
restore 1084
hasNodesLockedByOtherUser 1085
lock 1085
forceLock 1086
unlock 1086
forceUnlock 1087
isLocked 1088
getLockInfo 1088
setComment 1089
OKMNote 1090
Basics 1090
Methods 1090
add 1090
get 1091
delete 1091
set 1092
list 1093
getNoteHistories 1093
OKMNotification 1095
Basics 1095
Methods 1095
subscribe 1095
unsubscribe 1096
getSubscriptors 1096
notify 1097
OKMProperty 1098
Basics 1098
Methods 1098
addCategory 1098
removeCategory 1099
addKeyword 1099
removeKeyword 1100
setSigned 1101
isSigned 1101
OKMPropertyGroup 1103
Basics 1103
Methods 1103
addGroup 1103
removeGroup 1105
getGroups 1106
getAllGroups 1107
getUserGroups 1107
getProperties 1108
setProperties 1108
getPropertyGroupForm 1110
hasGroup 1111
getSuggestions 1112
registerDefinition 1113
getRegisteredDefinition 1113
getGroups 1114
getGroupProperties 1115
getProperties 1115
getSuggestBoxKeyValue 1116
getSuggestBoxKeyValuesFiltered 1117
OKMRecord 1118
Basics 1118
Methods 1118
create 1118
createSimple 1119

getProperties 1119
delete 1120
purge 1120
rename 1121
move 1122
copy 1122
copy 1123
extendedCopy 1124
isValid 1125
getPath 1125
getChildren 1126
setTitle 1126
setDescription 1127
createMissingRecords 1127
createFromTemplate 1128
createWizard 1128
OKMRelation 1130
Basics 1130
Methods 1130
getRelationTypes 1130
add 1131
delete 1132
getRelations 1132
getRelationGroups 1133
addGroup 1133
addNodeToGroup 1134
deleteGroup 1135
getGroup 1135
setGroupName 1136
getRelationGroups 1137
OKMReport 1139
Basics 1139
Methods 1139
list 1139
get 1140
execute 1140
OKMRepository 1142
Methods 1142
getRootFolder 1142
getTrashFolder 1142
getTrashFolderBase 1143
getTemplatesFolder 1144
getPersonalFolder 1144
getPersonalFolderBase 1145
getMailFolder 1145
getMailFolderBase 1146
getCategoriesFolder 1147
purgeTrash 1147
getUpdateMessage 1148
getRepositoryUuid 1148
hasNode 1149
getNodePath 1150
getNodeUuid 1150
getAppVersion 1151
copyAttributes 1151
getMetadataFolder 1152
getChangeLog 1153
OKMSearch 1154
Basics 1154
Methods 1157
find 1157
find 1158
find 1159

findPaginated 1160
findPaginated 1161
findPaginated 1162
findPaginated 1164
findWithMetadata 1165
findWithMetadata 1166
findWithMetadataPaginated 1167
findWithMetadataPaginated 1169
findSimpleNodeBasePaginated 1170
findByQuery 1172
findByQuery 1172
findByQuery 1173
findByQueryPaginated 1174
findByQueryPaginated 1175
findByQueryPaginated 1177
findSimpleNodeBaseByQueryPaginated 1178
saveSearch 1180
updateSearch 1180
getSearch 1181
getAllSearches 1182
deleteSearch 1182
getKeywordMap 1183
getCategorizedDocuments 1184
getCategorizedFolders 1185
getCategorizedMails 1185
getCategorizedRecords 1186
getDocumentsByKeyword 1186
getFoldersByKeyword 1187
getMailsByKeyword 1187
getRecordsByKeyword 1188
getDocumentsByPropertyValue 1188
getFoldersByPropertyValue 1189
getMailsByPropertyValue 1190
getRecordsByPropertyValue 1190
findMailPaginated 1191
csvExport 1193
OKMStats 1195
Basics 1195
Methods 1195
getDocumentsByContext 1195
getDocumentsSizeByContext 1195
getFoldersByContext 1196
getMailsByContext 1196
getRecordsByContext 1197
OKMUserConfig 1199
Methods 1199
setHome 1199
getConfig 1199
Common classes 1201
ArchiveUtils 1203
Methods 1203
createZip 1203
createJar 1203
ExecutionUtils 1205
Methods 1205
runScript 1205
runJar 1205
runCmd 1206
FileLogger 1207
Methods 1207
info 1207
warn 1207

error 1208
FileUtils 1209
Methods 1209
getFileName 1209
getFileExtension 1209
FormatUtil 1211
Methods 1211
formatSize 1211
parseSize 1211
formatDate 1212
ISO8601 1213
Methods 1213
parseBasic 1213
formatBasic 1213
GeneralUtils 1215
Methods 1215
uiNotify 1215
MailUtils 1216
Methods 1216
sendMessage 1216
sendMessage 1216
sendDocuments 1217
PathUtils 1219
Methods 1219
getParent 1219
getName 1219
getContext 1220
PDFUtils 1221
Methods 1221
merge 1221
getNumberOfPages 1222
fillForm 1222
PrincipalUtils 1224
Methods 1224
getUser 1224
StackTraceUtils 1225
Methods 1225
isCallingMe 1225
toString 1225
TemplateUtils 1227
Methods 1227
templateExists 1227
replace 1227
LegacySrv 1229
executeSQL 1229
executeHQL 1229
DbSessionManager 1231
Methods 1231
getSystemToken 1231
ConfigSrv 1232
getString 1232
getText 1232
getBoolean 1233
getInteger 1233
getLong 1234

getHtml 1234
getPassword 1235
getList 1235
Database description 1237
Node types 1237
SQL example queries 1238
Mark documents to be indexed again 1238
Node version 1238
SQL example queries 1239
Node metadata properties 1241
Node categories and keywords 1242
SQL example queries 1243
Advanced samples 1245
Security Provider 1246
General description 1246
Properties file 1246
Custom Principal Adapter 1246
Step 1: Download 1246
Step 2: OpenKM configuration 1247
Step 3: Install the plugin 1247
Authentication Provider 1247
Step 1: Download 1247
Step 2: Install the patch 1247
Step 3: OpenKM configuration 1248
Set unique document name based on metadata 1249
Tab UI content extension using KCenter methods 1253
Step 1 - create a sample HTML file 1253
Step 2 - Configure the external tab document extension 1253
Complete frontend extension sample 1255
Description 1255
Installation 1256
Implementation details 1257
index.jsp 1257
menu.jsp 1257
home.jsp 1257
Sample 1 1258
action.jsp 1258
distributor.jsp 1260
full_list.jsp 1261
Screenshot of sample1 view 1261
Sample 2 1262
action.jsp 1262
distributor.jsp 1263
add_form.jsp 1264
full_list.jsp 1265
Screenshot of sample2 view 1266

OpenKM 7.0 CE

Starting with version 7.0.0, the OpenKM Community version is available under a BSD-style license, with no source code provided.

OpenKM is an Electronic Document and Records Management System (EDRMS). Much of the literature about document management terms such as DMS, EDRMS, or CMS is usually influenced more by marketing considerations than by objective reasons.

In most cases, the differences between DMS, CMS, and EDRMS are subjective or irrelevant from a document management point of view. The term CMS has had some marketing success and is most commonly used. However, we prefer the term **EDRMS**, perhaps with less success in the marketing world, but is 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 a proper audit trail.
- When needed, apply legal record holds.
- Manage records disposition.
- Preserve records throughout their life cycle.

Some of OpenKM's goals are:

- Increase **efficiency: faster** document approvals, reduced manual data entry, and automated recurring tasks.
- Set **compliance** levels. The system helps organizations avoid penalties when regulators, auditors, or government authorities conduct 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 is impossible. OpenKM can solve complex day-to-day scenarios; it is not a secure application, but it has many features that can be used and configured in several ways.

Please take the time to read the documentation; it is time well spent. If you have any doubts, contact us. Usually, a problem has several ways to be resolved.

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

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 them to real-case scenarios.

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

Hardware and software requirements

This section presents information about the environment needed to run an OpenKM server, listing the required and optional software and providing facts about their configuration.

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

For best performance, OpenKM stores the binary content of the document in the filesystem. So, in case you have a server for the database and another for the application itself, keep in mind 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 with OpenKM quite well, and is conveniently priced.

The only requirement to run the OpenKM is to have **Java 8** (also called Java 1.8) installed.

Ensure you're installing OpenJDK 1.8; if you install JRE, the application will not run correctly.

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:

```
Java version "1.8.0_31."
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)
```

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?).

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

Installing Java

For Windows users

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

1. Download it from the Oracle website (Java 8) and choose the right platform for your hardware and Windows version.
2. Run the downloaded .exe file and follow the instructions displayed.

For Linux users

You can download Java 8 from the Oracle website (choose the right architecture). You can find RPM and TGZ packages to install. If you use Ubuntu, the following is a better method:

```
$ sudo add-apt-repository PPA:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
```

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
```

For Mac OS X Users

Java packages and instructions for installation are available from the Oracle website:

- Download
- Instructions

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 (at least 2 GB is required) and a fast hard disk such as SATA or better.

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

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

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

64-bit architecture is highly recommended.

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

We suggest having a modern server with good performance.

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

For more information about hardware requirements, please consult with your sales & marketing representatives, as they should provide you with a hardware requirements estimation.

(See the questionnaire at the bottom of the page)

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

Browser support

OpenKM user interface is based on the Node.js framework, which means it supports modern browsers.

Current browsers supported:

- Firefox
- Latest versions of Internet Explorer
- Chromium and Google Chrome
- Latest version of Opera

We encourage you to update your browser to the newest version.

We recommend Firefox and Chrome rather than IE for several reasons: better performance, fewer security holes, and better standards compliance.

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 a Linux distribution and build the packages for it. However, any recent Linux distribution should be usable as long as the:

- OpenJDK 8 is available.

- You can adapt the provided Linux scripts.
- Third-party Third-party software requirements

Windows (Known working configurations):

- For development: Windows 7, Windows 8
- **For deployment: Windows Server 2003, Windows Server 2008, Windows Server 2012**

Databases

Known working configurations:

- MySQL
- Oracle
- PostgreSQL
- SQL Server

The application internally uses Hibernate; in fact, you should get it running with any supported Hibernate dialect. For more information, read Hibernate: SQL Dialects.

JVM

Although OpenKM can run with other JDKs, we recommend using OpenJDK 8 (64 bits 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 giving them as much detailed information as possible for a more accurate estimation using this basic set of questions:

- Estimated number of concurrent users.
- Type of users (% of readers and % of writers).
- Estimated first-year number of files.
- Estimated annual growth in the number of files.
- Estimated average document size.

- Number of documents in the initial import
- Are these documents uploaded by users or by an automatic process?
- Most common document types (MS Office, OpenOffice, PDF, etc.).

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

- Do you have an initial import of documents?
    - If so, how many?
    - If so, what is the size in GB of the data?
- How many records are created in a day?
- How many documents are created in a week?
- How many documents are created in a month?
- How many documents are created 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 types like video or large TIFF files, etc.?
    - Describe them, especially those that are uncommon in most companies.

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

Third-party software requirements

OpenKM takes advantage of using third-party software for some features. Some of this software needs to be installed on the server; others are optional.

Software	Usage	
LibreOffice	• Converts documents to PDF. • It is needed for the preview feature.	
ImageMagick	• Transforms images. • Generates thumbnails. • It is needed for the preview feature.	Usually On Winc
Ghostscript	• Interpreter for the PostScript language and PDF. • Needed by ImageMagick.	Usually
Tesseract	• OCR engine. • It is used for the text extraction feature.	It is requ
FFmpeg	• Converts and streams audio and video.	
ClamAV or equivalent	• Antivirus	Required

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 compiled for your platform. But OpenKM also depends on third-party applications to be fully configured. For example, it needs a working LibreOffice installation to preview documents. Or you may 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, keep in mind that OpenKM does not need a graphical desktop environment. So, in Linux systems, you can avoid installing these packages and save some memory. Another essential piece of advice is to do a minimal Operating System installation. We only need the minimum to access the server. In the case of the Linux server, we prefer to use SSH because it is fast and secure. Please don't install Java, Tomcat, or any other software unless you have already asked us.

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

- How many documents do you want to store?
- How many users will access OpenKM?
- How many concurrent users?
- How much memory does the server have?
- How many CPUs does the server have?
- How much 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 distro (name and version)?
- Does the server have a GUI (X-Windows) installed?
- Is it a clean and minimal installation (only SSH), or have you installed any 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

	Linux Mint Debian Rocky Linux Oracle Linux Red Hat SUSE
Windows	Windows Server 2025 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 an LTS Server 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 20.04
- Ubuntu 22.04
- Debian 11
- Debian 12

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

If you are using 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 8 is still not installed, please install the following:

```
$ apt install openjdk-8-jre
```

If Java 8 packages are not available in your distro repositories, you need to download and install them from Eclipse Temurin.

OpenKM currently only works with Java 8, so once the installation process is completed, verify the default version is Java 8:

```
$ java -version
openjdk version "1.8.0_352"
OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~22.04-b08)
```

```
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
```

Otherwise, set Java 8 as the default:

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

If you do not have any other preference for a database, we suggest installing MariaDB:

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

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

```
$ systemctl status mariadb
```

Installing OpenKM

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

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

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

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

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

```
$ 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:

- How can I install Java 8 on Debian 11 (Bullseye)?
- Install Java JDK and JRE on Debian 8

Post-installation checklist

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

- Verify the default Java is Java 8.
- 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, be aware that you should have a licensed Red Hat version; otherwise, you won't be able to install updates.
- Red Hat ISO can be downloaded from https://developers.redhat.com/products/rhel/download for testing, but you must register first.
- Remember that **CentOS is deprecated**, so you should switch to CentOS Stream.

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

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

In case of using a proxy to access the Internet, please look at 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 8 is still not installed, please install the following:

```
$ yum install java-1.8.0-openjdk
```

If Java 8 packages are not available in your distro repositories, you need to download and install them from Eclipse Temurin.

OpenKM currently works only with Java 8, so once the installation process is completed, verify that the default version is Java 8:

```
# java -version
openjdk version "1.8.0_352"
OpenJDK Runtime Environment (build 1.8.0_352-b08)
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
```

Otherwise, set Java 8 as the default:

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

If you 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 in case of using a proxy to access the Internet.

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

```
$ java -version
```

Otherwise, set it as the default:

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

Once verified, start the required services:

```
$ systemctl start xvfb openkm
```

You can verify the status of these services by running:

```
$ 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 that the default Java is Java 8.
- 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 the 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

Check that you have enough disk space.

We recommend installing these tools:

- 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:

```
java version "1.8.0_312"
Java(TM) SE Runtime Environment (build 1.8.0_312-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.312-b07, mixed mode)
```

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

We also recommend setting the JAVA_HOME variable:

```
setx JAVA_HOME "C:\Program Files\Java\jdk1.8.0_312" /M
```

Install Java

Do these steps only if Java 8 is not installed on your server.

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

1. Download it from the Oracle website (Java 8) and choose the appropriate platform for your hardware and Windows version.
2. Run the downloaded .exe file and follow the instructions displayed.

Installing database

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

Install MySQL if you do not already have it installed:

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

If you do not have MySQL client software, installing MySQL Workbench is also a good idea.

Check InnoDB is the default MySQL engine

```
show engines;
```

It should show something like this:

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

If it does not default, modify **c:\ProgramData\MySQL\MySQL Server 5.6\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-8.5.69.zip to c:\

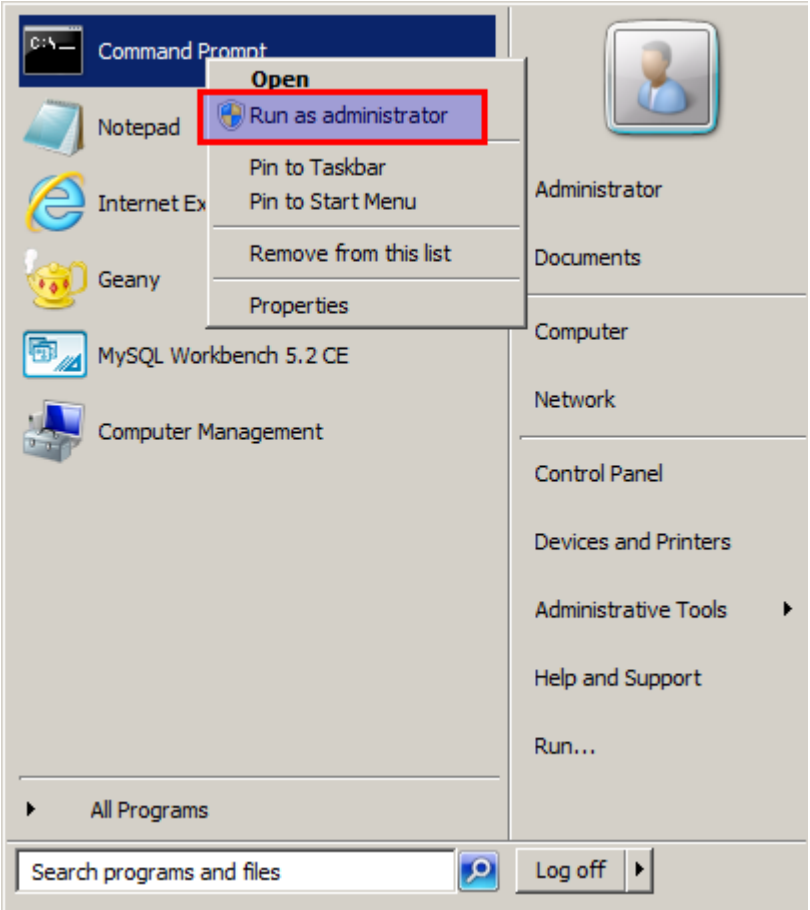
The Tomcat version numbering may change when upgrading to a major version.

Copy files from **C:\tomcat-8.5.69\bin\x86** or **C:\tomcat-8.5.69\bin\x64** to **C:\tomcat-8.5.69\bin** based on the server architecture.

Configuring Tomcat as a service

Register for the service:

Open a Windows terminal as an administrator.




```
cd c:\tomcat-8.5.69\bin
service.bat install
```

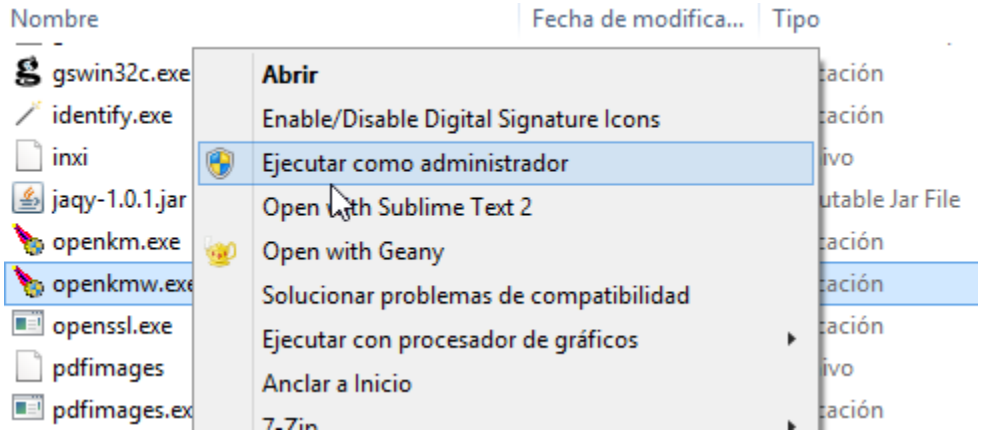
Set the JVM parameters for the Tomcat service:

```
cd c:\tomcat-8.5.69\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"
```

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

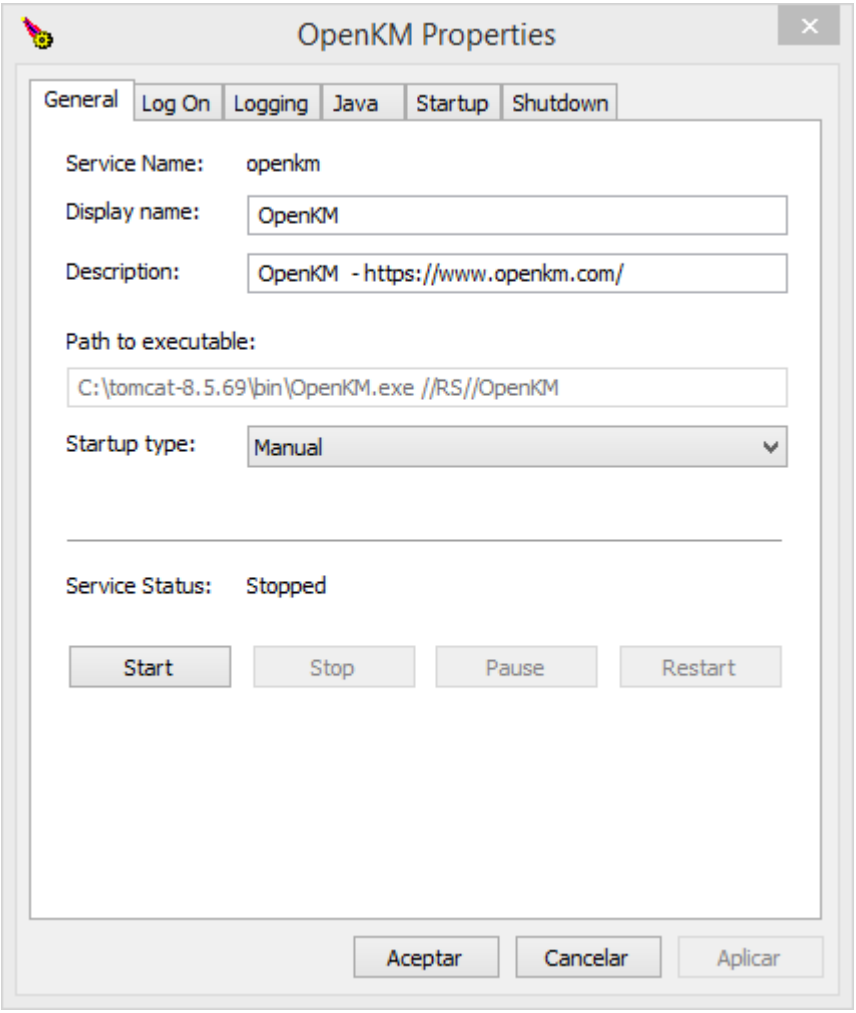
```
openkm //US//OpenKM ++JvmOptions="-Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
```

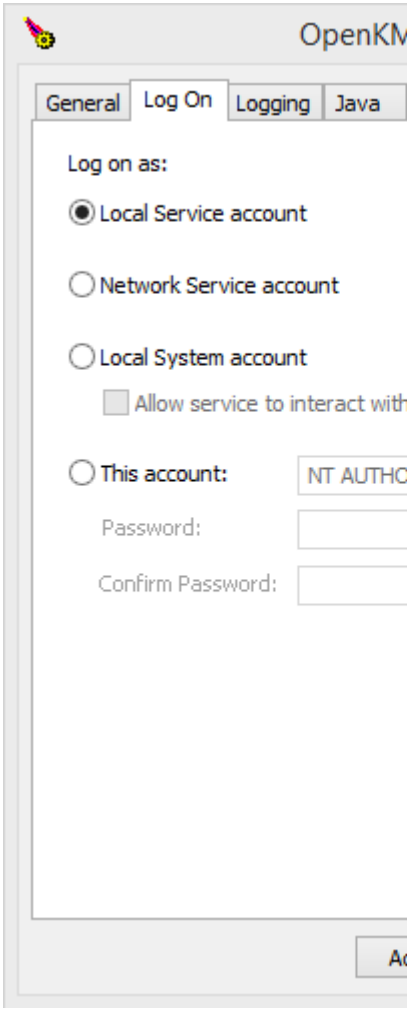
Run openkmw.exe as an administrator:

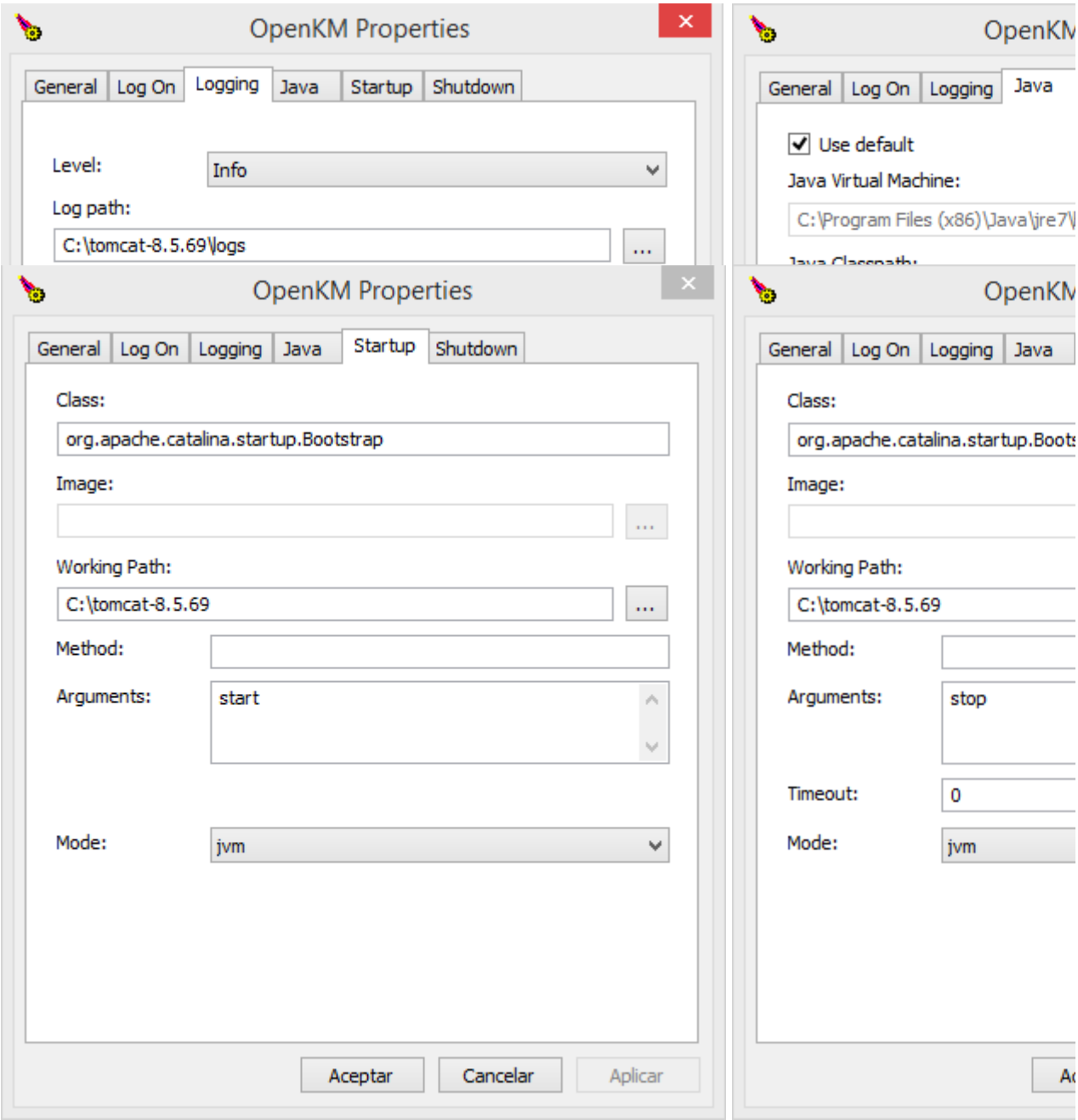


Check the configuration parameters:

> 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


General
Log On
Logging
Java
Log on as:
Local Service account
Network Service account
Local System account
Allow service to interact with
This account:
NT AUTHO
Password:
Confirm Password:


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

More information about how to configure the Tomcat service on Windows at: http://tomcat.apache.org/tomcat-8.5-doc/windows-service-howto.html

Check the service

From terminal

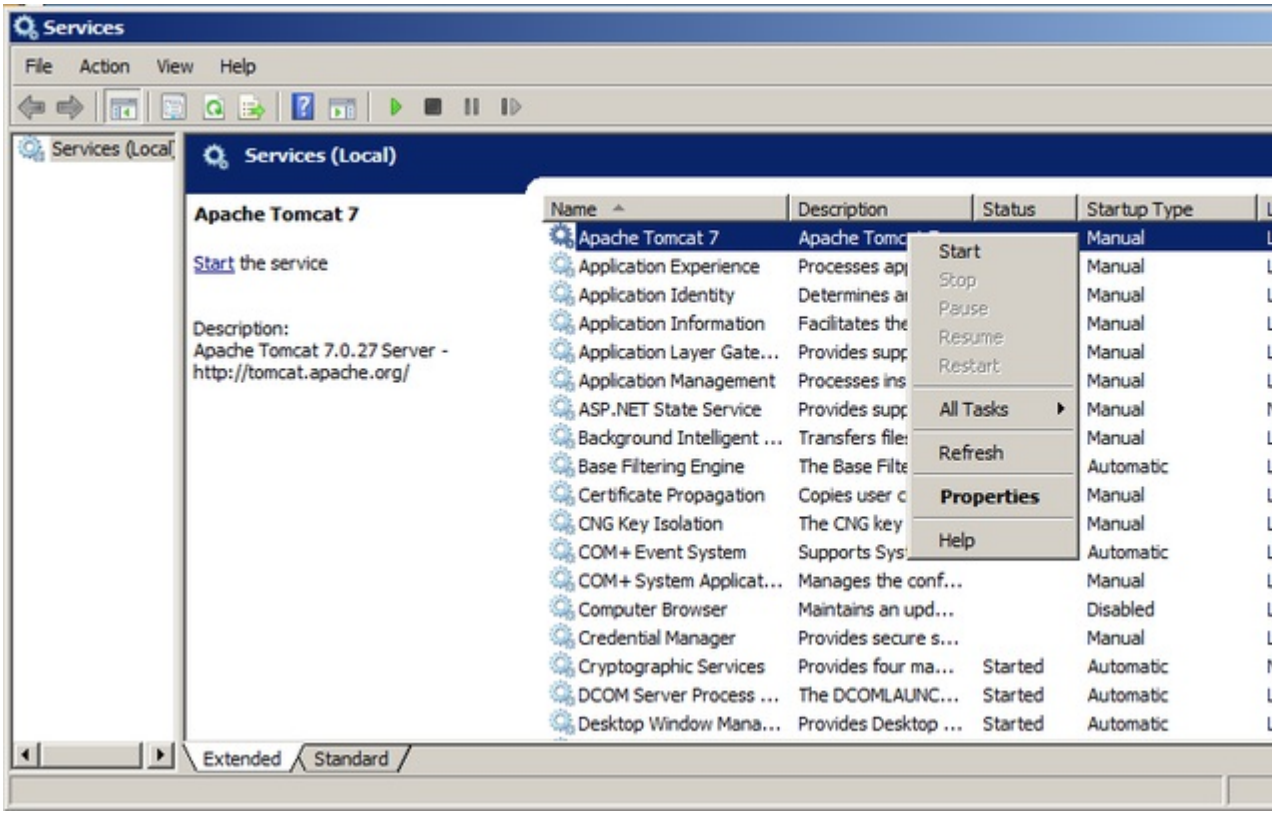
Start the service:

```
$ net start OpenKM
```

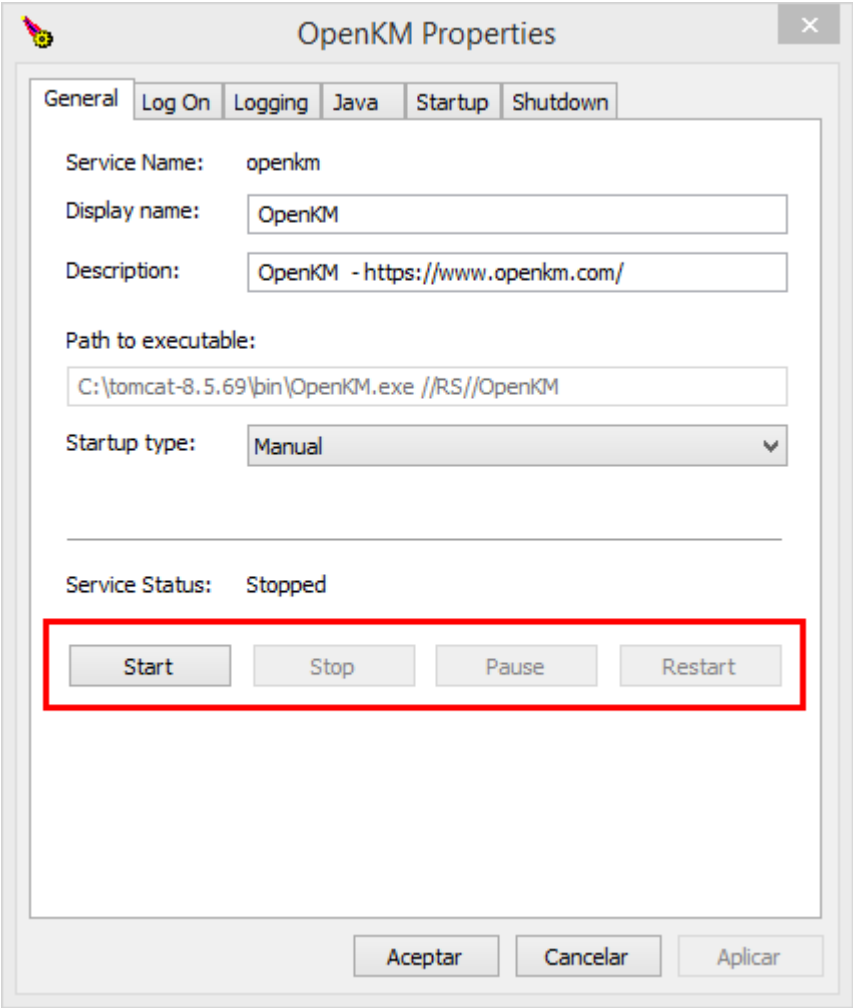
Stop the service:

```
$ net stop OpenKM
```

From the Windows service panel



From openkmw tool



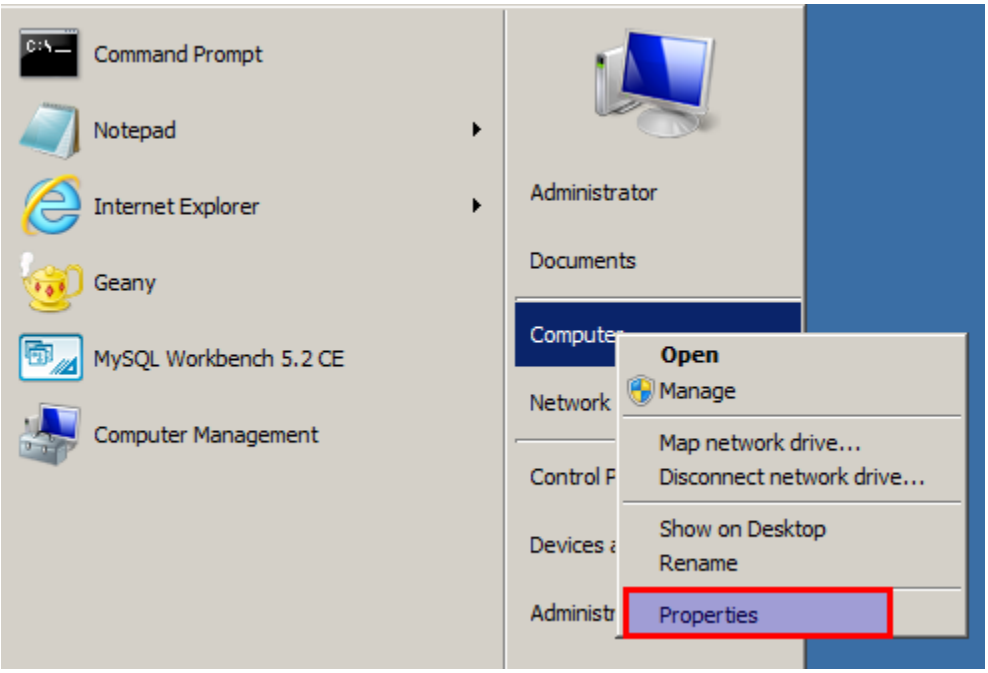
Configuring environment variables

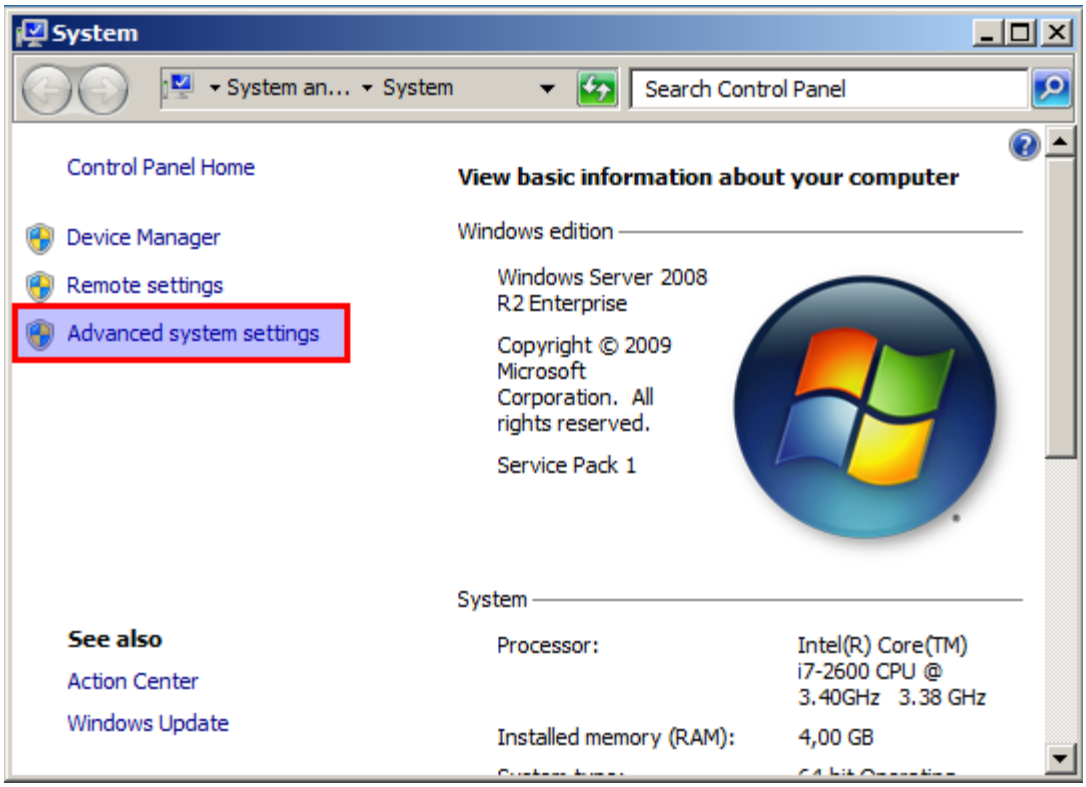
Add to the environment path variable the **bin** folder:

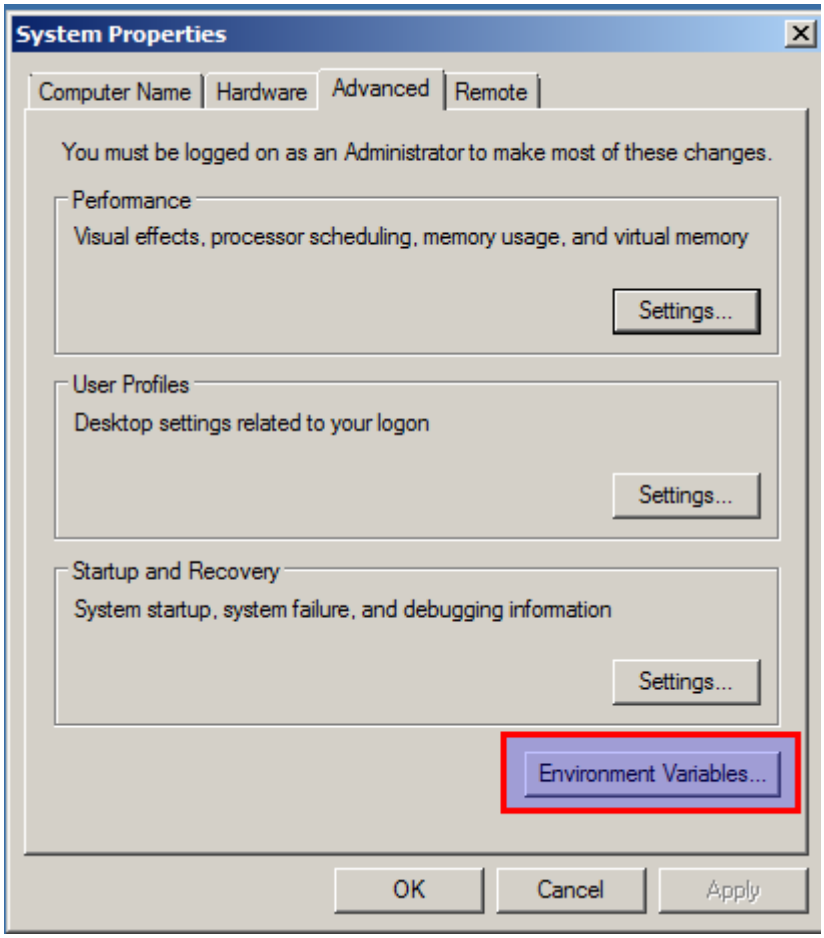
```
setx PATH "%PATH%;C:\tomcat-8.5.69\bin" /M
```

These operations can also be done from Windows environment variables:

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


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


System
System an... System
Search Control Panel
Control Panel Home
View basic information about your computer
Device Manager
Remote settings
Advanced system settings
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-8.5.69/webapps

```
$ copy openkm.war c:\tomcat-8.5.69\webapps
```

Install third-party software:

Software	Required	Description
LibreOffice	Yes	We recommend using LibreOffice rather than OpenOffice because it has better compatibility 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 conversion 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 require 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.
An **Extras.zip** file is also 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.openoffice.path=C:/Program Files (x86)/LibreOffice
```

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

First-time application start:

```
$ net start OpenKM
```

Use the mTail program - or an equivalent - to monitor the c:\tomcat-8.5.69\log\catalina.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

Another way of installing OpenKM is by using the installer tool. This Java application simplifies 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 8. Please check these pages for more information about installing Java on your server; look for the **Checking Java version** section:

- Installing on Red Hat and CentOS
- Installing on Ubuntu and Debian
- Installing on Windows

The installer installs into the folder in which it runs, so before running, move the OKMInstaller.jar to the location where OpenKM should be installed. When running on Linux, it also changes the file's ownership 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.

```
Running on Linux: 1.3.78 (build: e339f3b)
Linux distro: linuxmint (Linux Mint - 19.1)
Local host: GL73-8RD (127.0.0.1)
Java version: 11.0.15
Free disk size: 65.0 GB
RAM size: 15.5 GB
CPU cores: 12
Server IP: 192.168.1.38
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 [@Ep8!ov7rTXE]:
---------------------------
- Database: mariadb
- Database host: localhost
- Database name: okmdb
- Database user: openkm
- Database password: @Ep8!ov7rTXE
- OpenKM version: 7.1.30
- Tomcat version: 8.5.69
---------------------------
Start install process? [y/N]:
```

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

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 running the application:
>
> ```
> $ java -Dhttp.proxyHost=192.168.0.1 -Dhttp.proxyPort=3128 -jar OKMInstaller.jar
> ```

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

Supported databases

The installer supports these databases:

- MariaDB (mariadb)
- MySQL (mysql)

- PostgreSQL (postgresql)
- SQL Server (sqlserver)
- Oracle (oracle)

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 that your code will run.

There is a public Docker image available at https://hub.docker.com/repository/docker/openkm/openkm-ce and it is the easiest way to get OpenKM running on your server.

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

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

```
$ docker run --name openkm-ce -p 8080:8080 openkm/openkm-ce
```

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

And 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 openkm/openkm-ce:7.0.1
```

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 we can say 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 openkm/openkm-ce:7.0.1
```

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's password.
DATABASE_DIALECT	The Hibernate database dialect.

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

Docker Compose

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

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
```

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

This is another sample Docker Compose configuration:

```
version: '3.2'

services:

  openkm_mysql:
    image: openkm/openkm-ce:7.0.1
    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 describes how to deploy OpenKM in a pristine Apache Tomcat.

Preliminaries

Take a look at the installation instructions for your operating system:

- Installing on Red Hat and CentOS (Manual)
- Installing on Ubuntu and Debian (Manual)
- Installing on Windows

Installing Tomcat

Download the latest Tomcat version from the Apache Tomcat page (Tomcat 8) 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 https://sourceforge.net/projects/openkm/files/common/

- 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** on Linux or **setenv.bat** on Windows. This file should be placed in **TOMCAT_HOME/bin**. This is the content for Linux:

```
JAVA_OPTS="$JAVA_OPTS -Xms512m -Xmx2048m -XX:+UseG1GC -XX:+UseStringDeduplication"
JAVA_OPTS="$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 for Windows:

```
set JAVA_OPTS=%JAVA_OPTS% -Xms512m -Xmx2048m -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 Red Hat and CentOS (Manual)
- Installing on Ubuntu and Debian (Manual)
- Installing on Windows

Start the application

Check the 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.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/okmdb?useUnicode=true&characterEncoding=UTF8&serverTime
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.MySQL57InnoDBDialect

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

This configuration works if your database is MySQL. If you are using a different database, please check the documentation for that database.

First-time application start:

```
$ start openkm
```

Use the mTail program (or equivalent) to watch the TOMCAT_HOME\log\catalina.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** with 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 Hikari Pool

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

Configuring MySQL and MariaDB

MariaDB is a community-developed fork of MySQL intended to remain free under the GNU GPL. Being a fork of a leading open-source software system, it is notable for being led by the original developers of MySQL. It was forked because of concerns about its acquisition by Oracle.

Preliminaries

Perform all changes with the application stopped.

Check if your MySQL installation has the InnoDB engine enabled:

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

It should show something like this:

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

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

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

Change default engine

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

and under [mysqld], add

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

After changing the default storage engine, the service must be restarted for the change 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

```
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, the database server needs to be restarted.

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 command:

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

More information at [MySQL: Case Sensitivity in String Searches]

Configure your openkm.properties

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

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

Edit the file $TOMCAT_HOME/openkm.properties

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

MySQL Timezone

If you want to set your timezone in your JDBC connection according to the timezone configured in your system, 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:

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

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

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

Configure Tomcat data sources

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

MySQL:

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

MariaDB:

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

MariaDB can use these driverClassName values:

- com.mysql.jdbc.Driver
- org.mariadb.jdbc.Driver

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

Configure the values of the following attributes:

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

It can only use a resource called JDBC/OpenKMDS.

Configure application login

Edit the file $TOMCAT_HOME/openkm.properties

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

Run application

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

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

Remove and create MySQL service in Windows.

> *Ensure your MySQL\bin folder is in your PATH 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 the 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, while in Linux they 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
```

MySQL tuning

Take this documentation as-is; it is not a definitive guide to tuning MySQL. You should read a lot of information and guidance before changing the MySQL configuration. The documentation has been created 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 @2.93GHz*
- *RAM: 7GB*
- *JVM settings for Tomcat: -Xms 3072 -Xmx 3072 -XX:PermSize=256m -XX:MaxPermSize=512m*
- *The repository is 1GB*

MySQL my.cnf changes

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

Using the MySQL server 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***

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

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

Before changing these values, please read 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 1GB, so 1792M is enough RAM for* ***innodb_buffer_pool_size****. The larger the repository, the more memory you need to allocate to* ***innodb_buffer_pool_size****. On an x86 MySQL server, you cannot allocate more than 2GB 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*

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

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

Results

Before optimization, the login time was about 45-55 seconds, with high disk I/O usage. After the MySQL optimization, logging in takes about 10 seconds.

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

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 destination folder:

```
$ 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 location 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,
...
```

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

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

to

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

Start MySQL service:

```
$ service mysql start
```

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

CentOS and RHEL

Stop the service:

```
$ service mysqld stop
```

Create a new destination folder:

```
$ 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 location 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
$ semanage fcontext -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 destination folder:

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

Optionally modify the location 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
```

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

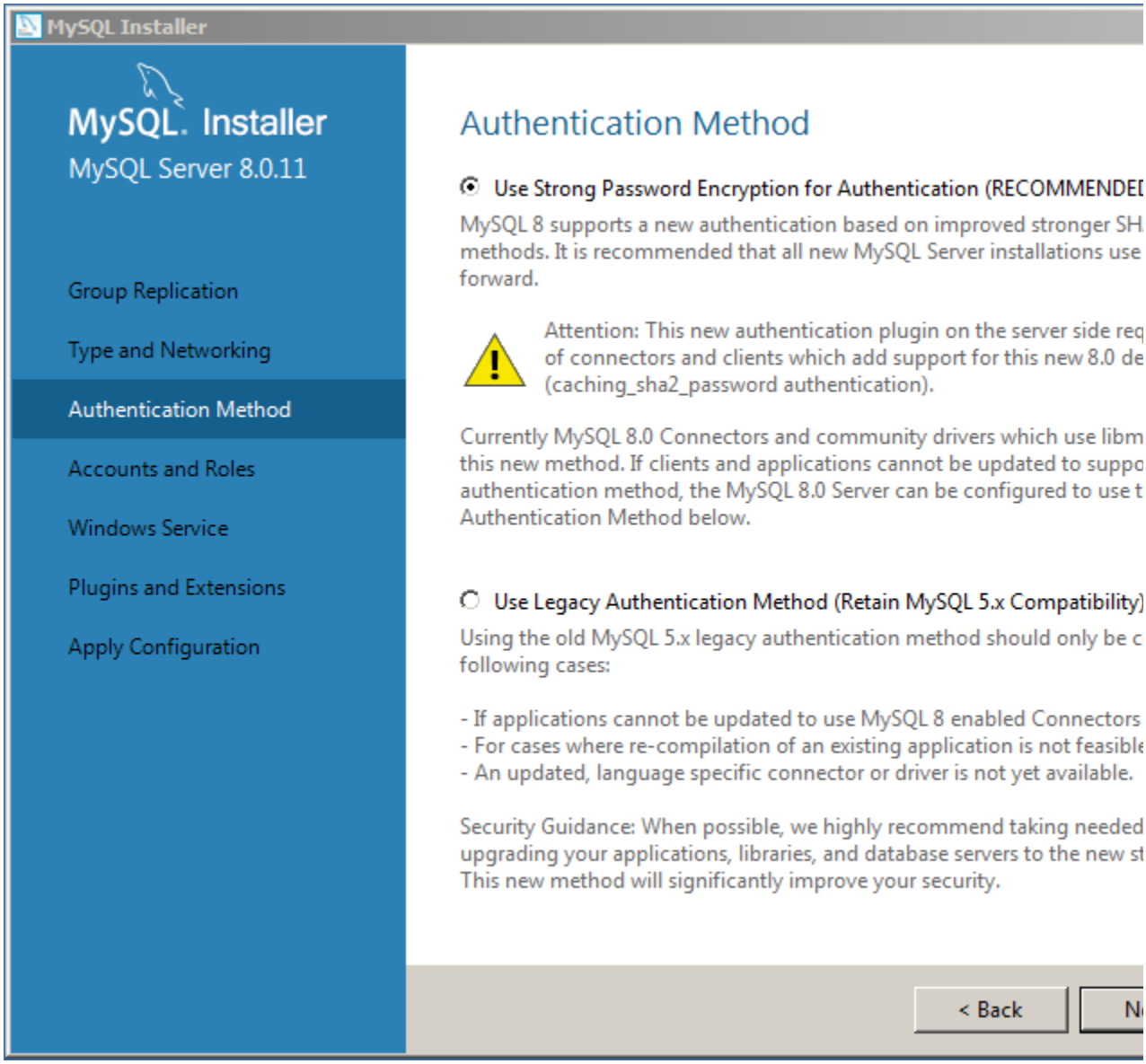
MySQL 8 troubleshooting

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 that 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 need to consult 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 if you decide to go ahead with SSL configuration:

- *https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-reference-using-ssl.html*
- *https://dev.mysql.com/doc/refman/8.0/en/creating-ssl-rsa-files-using-mysql.html*

*In Windows, it is relatively easy to enable the "**Legacy Authentication Method**":*

- *Execute the installer.*
- *Click 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 specify 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 no
```

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&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 can't switch to another database 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 connections 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 find 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 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**.*

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

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 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 changing this property.

Edit the file $TOMCAT_HOME/openkm.properties.

The supported dialects are:

- com.openkm.db.dialect.PostgreSQL94Dialect
- com.openkm.db.dialect.PostgreSQL92Dialect
- com.openkm.db.dialect.PostgreSQL9Dialect

```
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
```

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

```
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 following attributes:

- *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 above

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

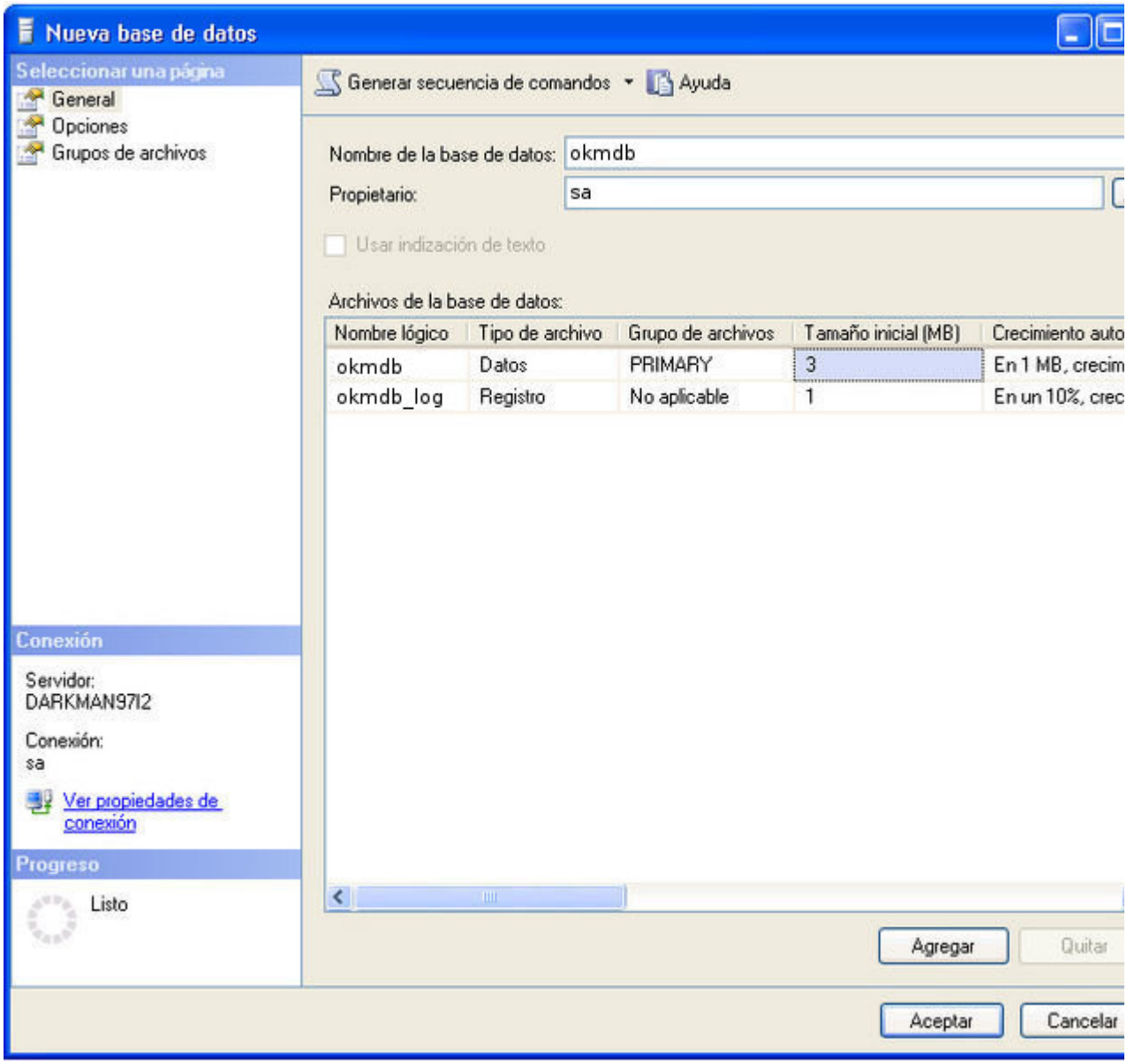
Read this related issue for more information PostgreSQL - ERROR o.h.tool.hbm2ddl.SchemaUpdate - could not complete schema update.

Configuring SQL Server

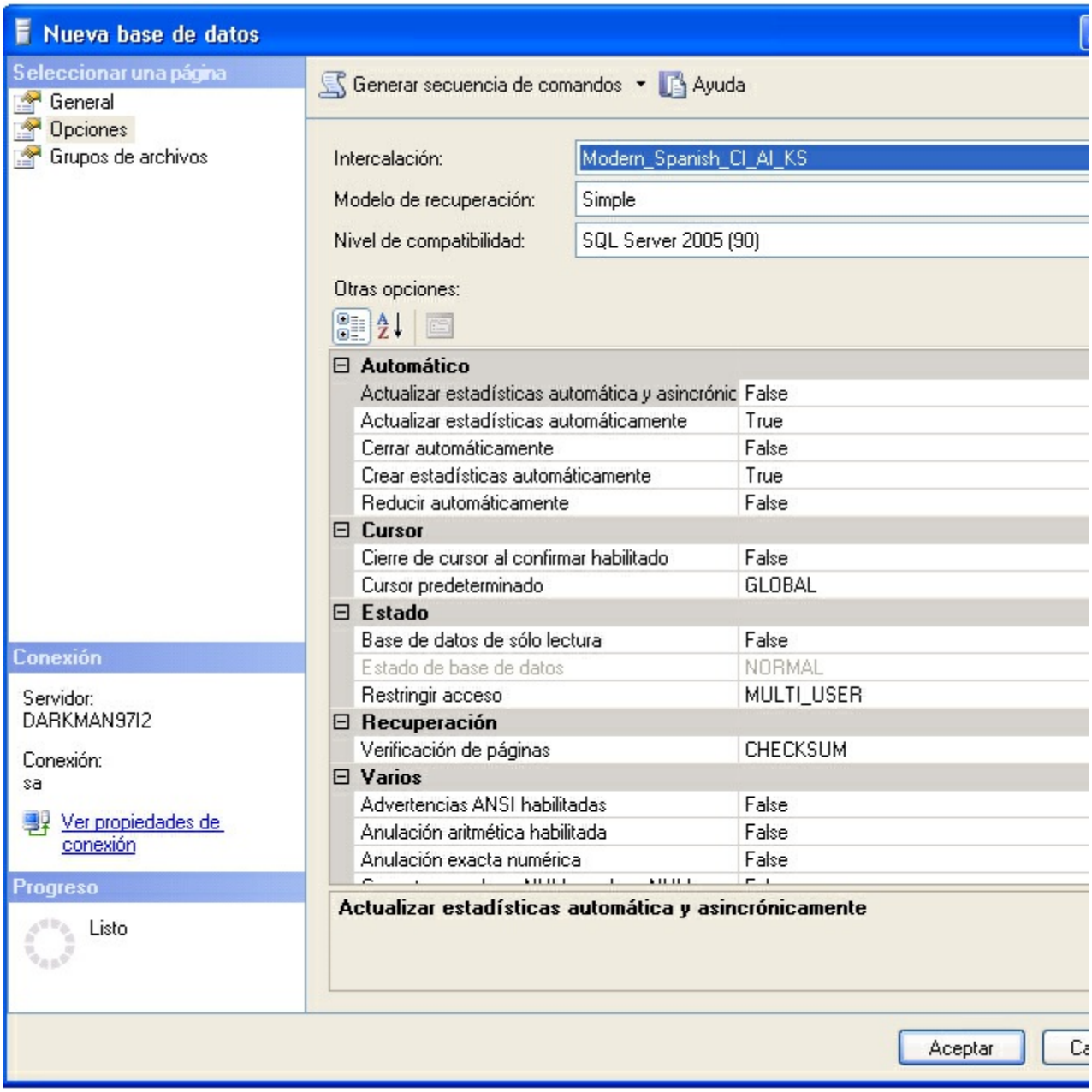
Database creation

Create a database and user

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



Select the appropriate database collation.



*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 collation should be set to **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, don't modify it because it can damage your installation.

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

database by changing this property.

Choose the Hibernate dialect that 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.SQLServerDialect*

Sometimes you may also need to set this compatibility level using the command line (ALTER DATABASE):

Alter database compatibility level

Edit the file $TOMCAT_HOME/openkm.properties

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

Configure Tomcat data sources

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

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

Configure the values of the attributes named:

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

Configure application login

Edit the file $TOMCAT_HOME/openkm.xml

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

Run application

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:

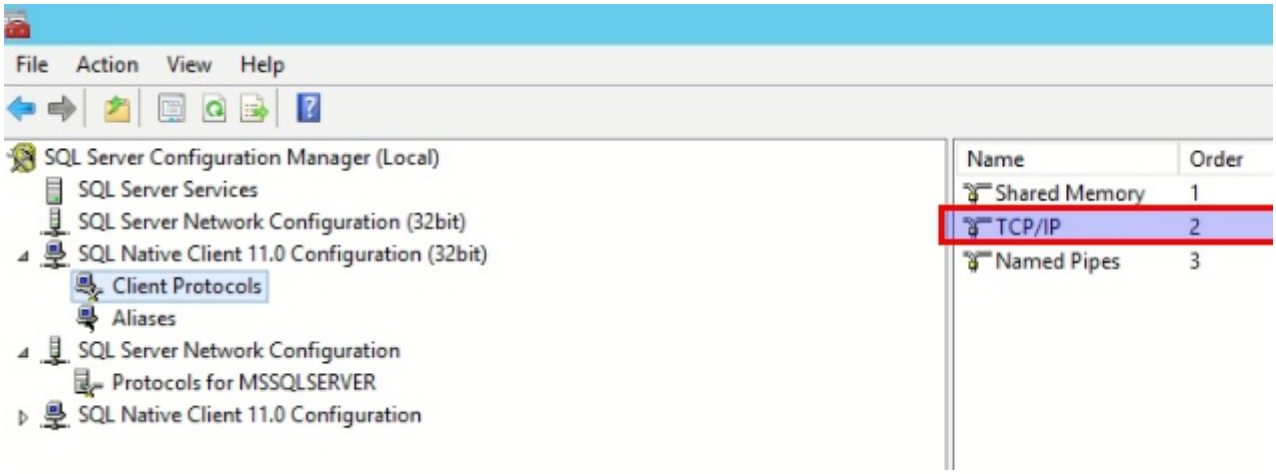
- *Microsoft SQL Server documentation*

Troubleshooting errors

Can't connect to SQL Server

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

In this section (first image), it may appear that TCP/IP protocol is enabled, but it is not. See the second image...



In this section you configure the TCP/IP protocol. Here you can see if 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: after several hours the connection is closed (it might be due to some extra configuration parameter in the latest SQL Server editions that is enabled by default or similar). There's a new configuration attribute named ***autoReconnect*** *that solves it.*

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

Unable 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.*

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

> *If you are experiencing 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

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

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

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

SQL Server alter compatibility level

To set SQL Server compatibility level 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 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 updated backup of your OpenKM.

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

> *OpenKM has several configurations, but for optimal performance, we recommend the following:*
>
> - *Use a default data store that saves binary information directly to the server disk (in a non-human-comprehensible format).*
> - *All 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. Starting with OpenKM version 7.0.x, the **Lucene indexes path location** in the file system **is not managed by "repository.home"**. For more information, take a look at the Change repository home.*

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

Backup steps:

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

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

An alternative to traditional backup methods is to export the whole repository from OpenKM Administration > Utilities >

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

Repository export and choose metadata and history options. However, we do not recommend this method for backups; we recommend 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

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

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 on Ubuntu, use this command to install it:

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

In the case of CentOS:

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

And execute the following 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:

```
$ 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 gawk imagemagick libreoffice tesseract-ocr
```

In the case of CentOS:

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

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

Windows detailed steps

You can follow these steps to 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/*

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 make a repository check. See Repository checker.

Backup scripts

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

Additional information

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

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

Linux backup and restore with BorgBackup

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

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

Installation

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

> *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 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 particular 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 be interested in unmounting 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, device 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 by 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

These scripts assume the following default values:

- ***$TOMCAT_HOME** value is "**/home/openkm/tomcat-8.5.69**"*
- *OpenKM **database** is named "**okmdb**."*
- *OpenKM **database password** value is "***secret***".*
- *OpenKM start & stop **service script** location is "**/etc/init.d/tomcat**"*

Sample script

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

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

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

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

These are the global script sections:

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

Configuration parameters explanation:

Parameter	Description
HOST	The server hostname.
DATABASE_PASS	The database password. *The database user who performs 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. 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-8.5.69"
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 by increasing 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 remote backup

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

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

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

*Also, 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 PostgreSQL backup

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

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

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

Configure crontab

To install the cron job, run the following:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your configuration:

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

More information at Crontab quick reference

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

Restoring backup

You can list the available backups:

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

Then decide to restore 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 latest backup

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

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

Sample for restoring a single folder (the /home/openkm/tomcat-8.5.69/ must exist)

```
rdiff-backup --restore-as-of now /mnt/backup/ms1/home/openkm/tomcat-8.5.69/ /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-8.5.69**: The backup of the Tomcat installation and OpenKM repository.*

Additional information

- *rdiff-backup home page*

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

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

Linux Sample Backup with Duplicity

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

Installation

Duplicity can be installed in Debian / Ubuntu as follows:

```
$ 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/RedHat 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-(
```

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:

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

```
$ duplicity --no-encryption --file-to-restore tomcat-8.5.69/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-8.5.69/openkm.properties plus the directory on which we based our backup (/home/openkm) equals /home/openkm/tomcat-8.5.69/openkm.properties

Preliminaries

These scripts assume the following default values:

- *The $TOMCAT_HOME value is "**/home/openkm/tomcat-8.5.69**"*
- *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 backs up to USB and the application database is MySQL.

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

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

These are the global script sections:

- *Global variable configuration under **## BEGIN CONFIG ##** section.*
- *Checking for 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 application (optional section) under the **# 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 application (optional section) under the **# Start tomcat** section.*
- *Show statistics (optional section) under the **# Status** section.*

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

- *Unmount disk (optional section) under the **# Umount disk** section.*

Explanation of configuration parameters:

Parameter	*Description*
HOST	*The server host name.*
DATABASE_PASS	*The database password.* *The database user 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	*Backup destination.*

*Create a backup script at **/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-8.5.69"
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 to increase the parameter value:*

> *The "**1M**" value indicates that backups older than one month will be removed. Changing it 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.*

```
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 handles incremental changes. Those files won't be deleted because, even though the backup may be older than 7 days, there are incremental backups that are younger 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 a full backup exists that is newer than them.

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 the **duplicity** documentation and Duplicity Backup Howto 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-8.5.69**: The backup of the Tomcat installation and the OpenKM repository.*

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

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 display directory contents and copy files, optionally using compression and recursion.

Preliminaries

In these scripts, the following default values are assumed:

- *The **$TOMCAT_HOME** value is "**/home/openkm/tomcat-8.5.69**"*
- *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 performed by the root user.

The script below backs up to a USB drive 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 is under the **## BEGIN CONFIG ##** section.*
- *Check for the root user under the **# Check root** section.*
- *Delete older database dump under the **# Delete older local database backup** section.*
- *Mount disk (optional section) under the **# Mount disk** section.*
- *Stop 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 **# Create backup** section.*
- *Start application (optional section) under the **# Start tomcat** section.*
- *Umount disk (optional section) under the **# Umount disk** section.*

Explanation of configuration parameters:

Parameter	*Description*
HOST	*The server host name.*
DATABASE_PASS	*The database password.* *The database user that performs 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	*Backup destination.*
RSYNC_OPTS	*Rsync parameters.*

*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-8.5.69"
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
```

Simpler 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-8.5.69/temp/* --exclude=tomcat-8.5.6
```

*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-8.5.69/temp/* --exclude=tomcat-8.5.6
```

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

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

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

Do remote backup to SMB or CIFS

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

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

Do PostgreSQL backup

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

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

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

Configure crontab

To install the cron job, run:

```
$ sudo mkdir /root/logs

$ sudo crontab -e
```

And add these lines according to your personal configuration:

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

More information at Crontab quick reference

*If you want to be notified by mail, you should install "**postfix**" service 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 mounted directory to 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 which backups to restore; 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-8.5.69**: The backup of the Tomcat installation and the OpenKM repository.*

Additional information

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

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-8.5.69**"*
- *OpenKM **database** is named "**okmdb**".*
- *OpenKM **database password** value is "***secret***".*
- *The OpenKM start & stop **service script** is located at "**/etc/init.d/tomcat**"*

Sample script

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

The script below backups up to a USB device 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 local database backups under the **# Delete older local database backup** section.*
- *Stopping the application (optional) under the **# Stop tomcat** section.*
- *Cleaning logs (optional) under the **# Clean logs** section.*
- *Backing up the database to the filesystem under the **#Backup database** section.*
- *Backing up the repository and database dump to the backup destination under the **#Backup and purge old backups** section.*
- *Starting the application (optional) under the **# Start tomcat** section.*

Explanation of configuration parameters:

Parameter	Description
HOST	*The server host name.*
NOW	*Current date in the format 'dd-MM-yyyy'.*
DATABASE_PASS	*The database password.* *The database user to perform backups 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 ##
```

```
HOST=$(uname -n)
NOW=$(date +"%d-%m-%Y")
DATABASE_PASS="*secret*"
OPENKM_DB="okmdb"
OPENKM_HOME="/home/openkm"
TOMCAT_HOME="$OPENKM_HOME/tomcat-8.5.69"
DATABASE_EXP="$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
```

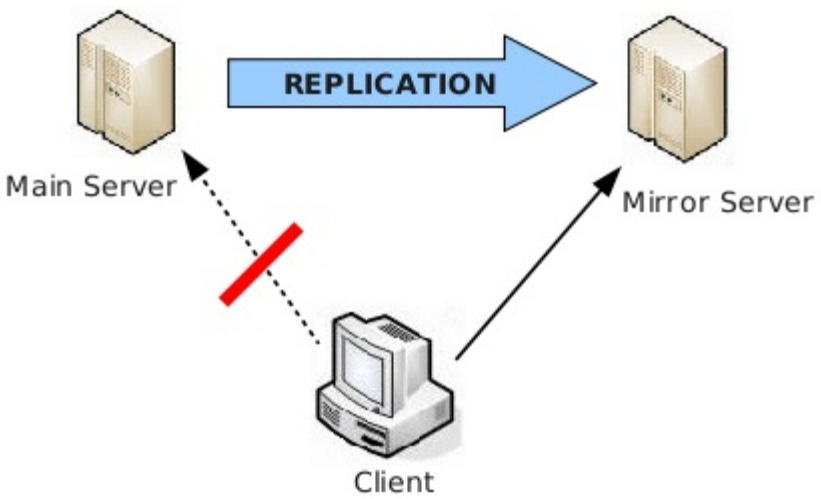
Add these lines according to your personal configuration:

```
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.*

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.



In this scheme, we assume:

- *Each node has its 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 becomes 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

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

- *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 to use an external file server (connected by NFS) so rsync is not required. A NAS solution improves data backup and availability and 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), or use providers like Oracle that offer specific database cluster solutions which should be taken into consideration.

Sample Linux implementation

The replication is done only in one direction, from the main server to the mirror server.

The mirror server must be configured to be in read-only mode so users can't add or modify documents.

The following script will propagate the repository changes from the main server to the mirror server:

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 daily 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-8.5.69"
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}:/h
```

```
# 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 file on the server.

```
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 having a backup. Refer to the Backup and restore section.

Linux

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

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 (the $TOMCAT_HOME folder). This user is usually named "**openkm**" (more info at Installing on Ubuntu and Debian (Manual)), but another may be chosen during installation.*
- *Update the OS.*

Stop the OpenKM service

It is important that, while moving the data between the two servers, no changes occur on the old server. That's why it is 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 sample below, 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 databases. 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 (Manual)).*
- ***Copy the okmdb.sql file to the new server****.*
- ***Restore the backup****.*

The sample below explains how to restore the MySQL and MariaDB databases. 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 located elsewhere. 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 at the destination*** *(otherwise, you will need to take extra configuration steps).*
- *Set* ***permissions for the owner of 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-8.5.69
```

The sample below shows how to set the owner (you need to execute this as root):

```
$ chown openkm:openkm -Rf /home/openkm/tomcat-8.5.69
```

Configure the service on the new server

- *Copy the file* ***/etc/init.d/openkm to the new server at the same location and register it as a service*** *(more info at Installing on Ubuntu and Debian (Manual)).*

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 (Manual)):

- *ImageMagick*
- *Ghostscript*
- *LibreOffice*

- *Tesseract OCR engine (remember to install the language packs required in your environment)*

Start the OpenKM service

Until the license is moved to the new server, you will only be able to access the administration user interface .

- ***In case the new server has less RAM than the old server, you need to check*** *the* ***$TOMCAT_HOME/bin/setenv.sh*** *file and adapt it properly.*
- *Check the openkm.log file for the startup process and look for possible errors.*

Additional steps

- *If the new server has a different number of cores, you need to check the value of the configuration parameters named "**text.extraction.threads**" and "**text.extraction.batch**".*
- ***Open administration*** *at http://localhost:8080/openkm/management/ and* ***execute a repository checker*** *(this action may take a long time to complete).*

Final checks

- *Check the preview of different document types (PDF, Microsoft Office).*
- *Check the thumbnails of different document types (PDF, Microsoft Office).*
- *Create a folder.*
- *Upload a new document.*
- *Download a previously existing document.*

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

Setup

*This section provides information for **configuring** and **tuning** OpenKM.*

The application contains 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 may also be interested in:

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

If you have an issue, 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 perform an antivirus check from the command line.

The example below is based on ClamAV, an open-source antivirus engine that detects trojans, viruses, malware, and 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 latest 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 a default minimal configuration (probably not suitable â€" you should configure the parameters for your situation), edit freshclam.conf and 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.*

- *Run* ***mkdir database****. If you run clamconf again, you will see that the database error is gone and that you have 0 signatures.*
- *Run* ***freshclam.exe****. This will start the initial signature download for ClamAV and may take a few minutes.*

At this point, you can run ***clamd.exe*** *to check that the server can run manually. (If you decide to test run at this point, you will have to kill the process for the next section)*

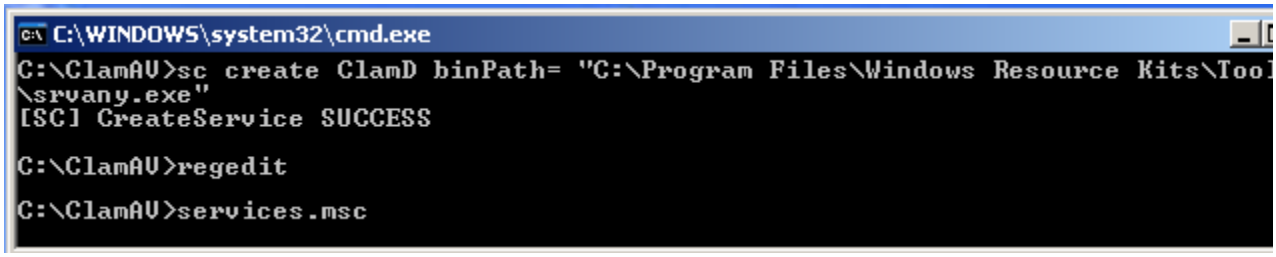
In case you need to create a freshclam.conf file, use this as a base:

```
# URL of server where database updates are to be downloaded from
# If this option is given multiple times, each will be tried in
# the order given until an update is successfully downloaded
#MAKE SYU
DatabaseDirectory "C:\Program Files\ClamAV-x64\db\"
DatabaseMirror database.clamav.net
# Number of times to try each mirror before moving to the next one
MaxAttempts 3
#LogFile C:\Program Files\ClamAV-x64\log\freshclam.log
#NotifyClamd C:\Program Files\ClamAV-x64\clamd.conf
LogFileMaxSize 20480000
LogTime true
UpdateLogFile C:\Program Files\ClamAV-x64\log\freshclam.log
```

Clamd Service Setup

Now to 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.

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.

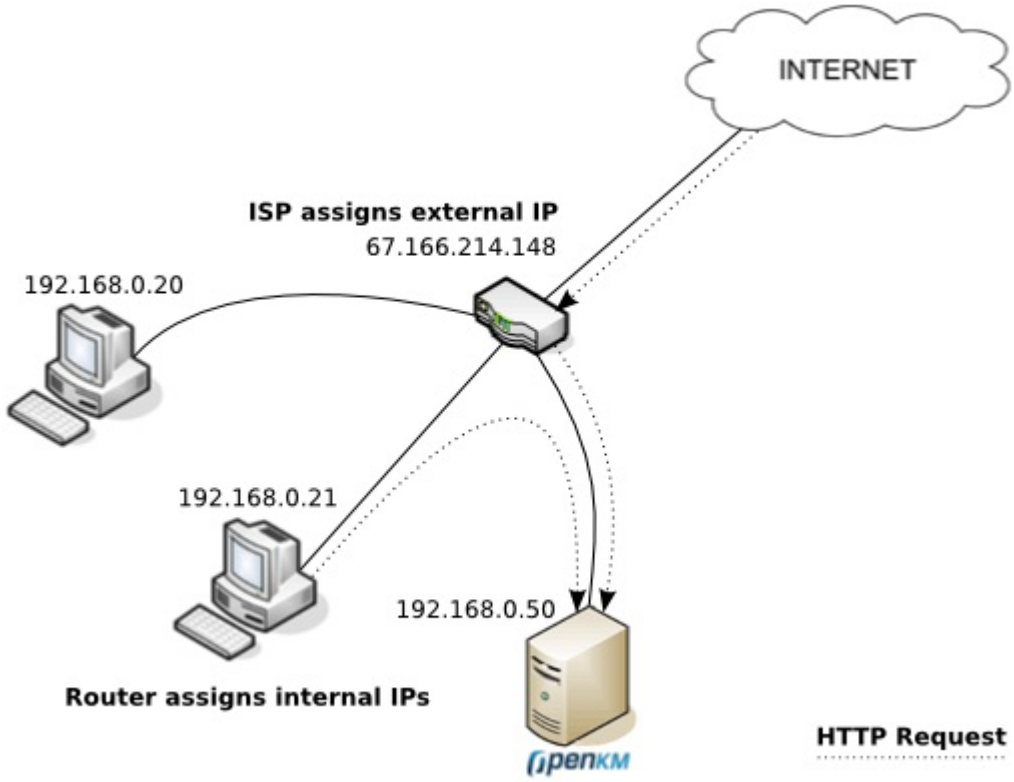
Internal IP vs external IP

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

The internal IP address (also known as "local IP address") is the address 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 sort of DNS server inside the LAN or modify every client host file to resolve to the internal IP.*

Debian and Ubuntu

The first thing is to install the required Apache software. 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 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/frontend/webSocket ws://127.0.0.1:8080/openkm/frontend/webSocket
  ProxyPassReverse /openkm/frontend/webSocket ws://127.0.0.1:8080/openkm/frontend/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/frontend/webSocket ws://127.0.0.1:8080/openkm/frontend/webSocket
  ProxyPassReverse /openkm/frontend/webSocket ws://127.0.0.1:8080/openkm/frontend/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 the log shows an error like ***Permission denied: proxy: AJP: attempt to connect****, to solve it execute the following command:*

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

If it works, make the change permanent so it persists across reboots:

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

Windows

The Apache web server is already included in the "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.

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

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 -->
<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.

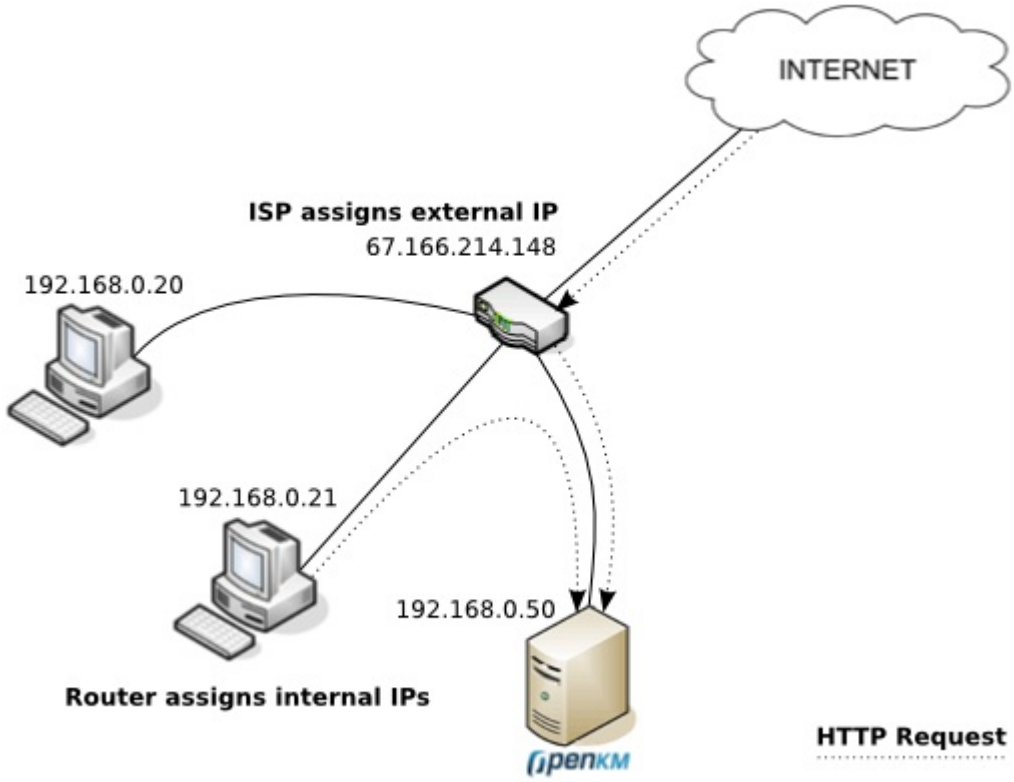
Internal IP vs external IP

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

```
$ vim /etc/apache2/sites-available/openkm.conf
```

```
<VirtualHost *:443>
  ServerName openkm.your-domain.com
  RedirectMatch ^/$ /openkm

  # WebSocket support - needs proxy-wstunnel
  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

  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 name.

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, 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 job to renew it automatically:

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

Also, keep in mind that both ports 80 and 443 must 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 name.

Finally, modify the ***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 set the configuration permanently 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 an 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.

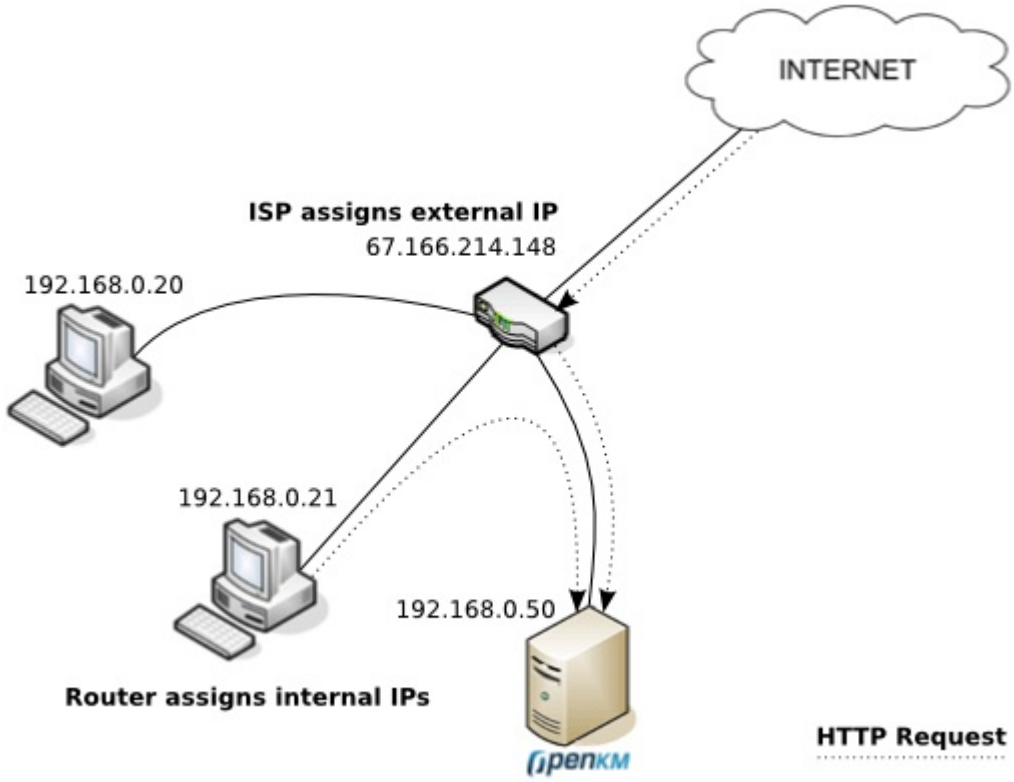
Internal IP vs external IP

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, 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:*

```
$ 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/frontend/webSocket {
       proxy_pass http://localhost:8080/openkm/frontend/webSocket;
       proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";
       proxy_set_header Host $host;
    }

    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 domain.

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 from http://openkm.your-domain.com/.

Another advantage of using Nginx is that you can log OpenKM access and generate web statistics.

Red Hat and CentOS

Use the yum application manager to install Nginx:

```
$ sudo yum install nginx
```

Enable it at boot:

```
$ sudo chkconfig nginx --level 2345 on
```

*Depending on your CentOS version, this Nginx version may be too old, so it's better to install it from their repositories. To do this, create a file called **/etc/yum.repos.d/nginx.repo** and include one 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:

```
$ 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/frontend/webSocket {
        proxy_pass http://localhost:8080/openkm/frontend/webSocket;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
    }

    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 domain.

Check the configuration

Restart Nginx:

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

Check access to your OpenKM installation from 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.

In case you see these kinds of errors in the Nginx log:

```
connect() to 127.0.0.1:8080 failed (13: Permission denied)
```

Execute this command and try again:

```
$ setsebool -P httpd_can_network_connect on
```

Here are several useful commands related to SELinux:

Get all related HTTP configuration settings

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

More information at SELinux Booleans.

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 at Lucene documentation site and Guide to Lucene Analyzers.

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

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

If you want only a white space tokenized analyzer, you can try this one WhitespaceAnalyzer.

If you have not configured the search analyzer before you first start OpenKM, 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 Lucene Indexes.

Once the operation has been completed, the Lucene indexes will use the new analyzer.

For more information, take a look at Rebuild indexes.

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

Configure an Analyzer

*Edit the **$TOMCAT_HOME/openkm.properties** file and add the line:*

```
spring.jpa.properties.hibernate.search.analyzer=org.apache.lucene.analysis.es.SpanishAnalyzer
```

The changes will take effect after restarting the application.

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

Configuring 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 info about installation on different OSs and Linux distros at https://tesseract-ocr.github.io/tessdoc/Installation.html

Tesseract is an open-source OCR engine adopted by Google. It works really well. It can natively read TIFF documents and has a high recognition rate with images of 300 dpi resolution converted to line art (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	*In CentOS you can install Tesseract from the EPEL repository:* *$ yum install epel-release* *$ yum install tesseract* *The default installed language is English, so depending on your locale you 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 right, a file named **text.txt** that 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} ${fileOut} -l esp`
	*The parameter **-l esp** indicates that Spanish language support must be used when executing OCR.*
	To see all Tesseract options, execute:
	`$ 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 Spanish language support must be used when executing OCR.*
	To see all Tesseract options, execute:
	`c:\Tesseract-ocr-3.0.2> tesseract.exe`

*Parameter **${fileIn}** will be replaced internally with the image document to be processed.*

*Parameter **${fileOut}** will be replaced internally with a temporary text file.*

*You can set additional parameters, such as the **-l spa** parameter in the example.*

Each parameter is separated by a single space.

*Once configured, please go to **Administration** > **Utilities** > **Plugins**, choose **com.openkm.plugin.extractor.TextExtractor** from the list 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, in upside-down scanned pages, where the standard OCR direction will fail.

When the feature is enabled, the operation will be done 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 on 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 the 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 a Certification Authority (CA)

Create the 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
```

Create public and private self-signed keys for our Certification Authority (in the example they are valid for 10 years). You will need to specify certificate data and a password to encrypt the key, 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. The key file (server.key) and the certificate request (server.pem) will be created.

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

The CA should sign the certificate request. Copy server.pem as the certificate request 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 only be readable 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:

```
$ 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 a pair of keys. You will need to specify a password and certificate data. A key (user.key) and a certificate request (user.pem) will be created:

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

The CA needs to sign the 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** file should be installed in the user's browser.*

Certificate revocation

If the server has been compromised, the CA should revoke the certificate. To revoke a certificate, find the serial number in /opt/ca/index.txt.

```
$ openssl ca -config /etc/ssl/openssl.cnf -revoke newcerts/<serial-number>.pem
```

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, convert it 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 and add SSLCARevocationFile:

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

Restart Apache service.

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

Configuring mail

The application sometimes sends emails to users, for example, when the users have subscribed to a document or folder.

By default, Tomcat mail is configured assuming your server has a mail service binding on your localhost.

These are the default values:

```
spring.mail.host=localhost
spring.mail.port=25
spring.mail.username=
spring.mail.password=
spring.mail.from=noreply@openkm.com
spring.mail.properties.mail.smtp.auth=false
spring.mail.properties.mail.smtp.starttls.enable=false
```

Change mail configuration

Get your mail account data (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, which means the user who appears in the "From" mail header.*

Configuring Tomcat memory utilization

It's also a good idea to configure the Tomcat memory utilization.

*Edit the file $TOMCAT_HOME/bin/setenv.sh and edit the parameter **JAVA_OPTS**, where you can increase the system memory managed by the JVM (Java Virtual Machine).*

OpenJDK 1.8

These are the default configuration file values. Tomcat will reserve 4096 MB (Xmx parameter), and the size of the heap since version 1.8 is dynamically assigned by the JVM. For this configuration, your server needs at least 5.5 GB of RAM (it is good practice to always 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.ignoreunknownencod
CATALINA_PID=$CATALINA_HOME/catalina.pid
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CATALINA_HOME/lib/sigar
```

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

Configuring Tomcat 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 are probably already 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.*

Configuring Tomcat session timeout

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

To change the Tomcat session timeout, edit $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 timezone

Tomcat works in 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 the timezone 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 **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

```
<?xml version="1.0" encoding="UTF-8"?>
```

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

```
<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:user-service>
        <security:user name="okmAdmin" password="admin" authorities="ROLE_ADMIN" />
      </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>
```

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

```
<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

```
<?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:user-service>
        <security:user name="okmAdmin" password="admin" authorities="ROLE_ADMIN" />
      </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 search highlighter

The search highlighter feature shows highlighted 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
```

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

It is not 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

Configuration below is based on setting proxy configuration parameters in the JVM environment. There's another option, which is configuring the proxy at the OS level. Here is one of several options:

*Edit your "****/etc/bash.bashrc****" file as root and put these lines at the end of your "****/etc/bash.bashrc****" file:*

```
export http_proxy=http://username:password@proxyserver.net:port/
export ftp_proxy=http://username:password@proxyserver.net:port/
```

- ***Stop OpenKM****.*
- ***Edit the $TOMCAT_HOME/bin/setenv.sh*** *file and add the lines:*

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

*The "****user****" value must be the user 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 used to connect 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.ignoreunknownenco
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 this configuration, which attempts to use Google DNS:

```
JAVA_OPTS="$JAVA_OPTS -Dsun.net.spi.nameservice.nameservers=8.8.8.8"
JAVA_OPTS="$JAVA_OPTS -Dsun.net.spi.nameservice.provider.1=dns,sun"
```

- ***Start OpenKM***.
- *Go to* ***Administration*** > ***Tools*** > *Scripting and execute the script below (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

Configuration below is based on setting proxy configuration parameters in the JVM environment. There's another option, which is configuring the proxy at the OS level; consider looking for this information on the Microsoft TechNet website.

Step one

Check 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 the host URL of your proxy.

*The "****user****" value must be the user 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 used to connect 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 this configuration, which attempts to use Google DNS:

```
set JAVA_OPTS=%JAVA_OPTS% -Dsun.net.spi.nameservice.nameservers=8.8.8.8
set JAVA_OPTS=%JAVA_OPTS% -Dsun.net.spi.nameservice.provider.1=dns,sun
```

- ***Start OpenKM from the command line**.*

```
c:\tomcat-8.5.69\bin\> catalina.bat run
```

- *Go to **Administration** > **Tools** > Scripting and execute the script below (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 everything went right in step one, you can now configure the parameters in the OpenKM service.

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

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

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/>"));
```

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.*
index	*No*	*The folder that 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 that contains the application's cached files. For example, to preview documents, it is sometimes necessary to transform the original document into another format. The conversion consumes a lot of hardware resources; for this reason, the application stores these converted files to be reused later. The contents 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 that contains all the binary data of the documents.* > *It is **mandatory to back up** this folder; otherwise, the binary data of the documents could be lost in a server hardware failure.*
extraction	*Yes*	*This folder contains the text extracted from those documents stored in OpenKM. A backup is not necessary, but it may save time.*

Move the 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

> *When the "**repository.home**" parameter is set, the **location** in the file system **of the Lucene index is not affected**. You should use the specific parameter named "**spring.jpa.properties.hibernate.search.default.indexBase**" to move the Lucene index 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.default.indexBase*	*${catalina.home}/repository/index*
tmp	*repository.temp.home*	*${repository.home}/tmp*
cache	*repository.cache.home*	*${repository.home}/cache*
datastore	*repository.datastore.home*	*${repository.home}/datastore*
extraction	*repository.extraction.home*	*${repository.home}/extraction*

> *These parameters can be used in combination with the parameter "**repository.home**".*

- *Stop the application*
- *Move the "**$TOMCAT_HOME/repository/{folder}**" to another location.*

- *Edit the "**$TOMCAT_HOME/openkm.properties**" file and add some properties.*
- *Start the application*

Sample

Windows

```
spring.jpa.properties.hibernate.search.default.indexBase=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
```

Linux

```
spring.jpa.properties.hibernate.search.default.indexBase=/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
```

Troubleshooting

The Windows mapped network disk is not working

We recommend that you do not use network storage (NAS or Samba) for storing the repository. Network issues can affect the repository, so we consider it a dangerous configuration.

*There seem to be restrictions when using network drives (mapped with the "**net use**" command) and applications running as services. To fix 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, rather than 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 have 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 the default roles to new names in the database

> *In the case of an **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 from 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, it is only necessary to insert it into this table.

Extensions that do not require additional software configuration:

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

```
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 ('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 ('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
```

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 the 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, and, once this section is configured, go to Spring Security configuration (openkm.xml changes).

Usually you only want to retrieve a subset of users and roles present in your LDAP, to be shown in the user interface or 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 at login*

Active Directory

- *Active Directory basic configuration*
- *Active Directory advanced configuration*
- *Active Directory example with referral enabled*
- *Active Directory example with login based on filtering users by roles (suggested).*
- *Active Directory example with remember-me services*
- *Active Directory mixed configuration*

Uncategorised examples shared by users

- *http://pastebin.com/iQa4LhyY (more information OpenKM public forum)*

Tools

- *Apache Directory Studio.*
- *Active Directory Explorer Utility.*

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

OpenLDAP example with URL base at login

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***
- *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.)*

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

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 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 **logging 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 value of the **principal.adapter** parameter 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 be empty.*

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

Field / Property	*Type*	*Description*
principal.adapter	***String***	*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*
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 the default filter base queries to **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 a parent.

- *User filter is uid={0}*
- *Finally, take into consideration the value 1 in 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

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*
- *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*

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

- *cn=user2,ou=organization1,dc=company,dc=com*
- *cn=user3,ou=organization2,dc=company,dc=com*
- *cn=user4,ou=organization2,dc=company,dc=com*

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

The OpenKM integration with LDAP has two steps. In the first step, configure OpenKM to retrieve the list of users and roles from LDAP. OpenKM caches this list 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 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 the next parameters are successfully configured, these lists will remain empty.*

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

Field / Property	*Type*	*Description*
principal.adapter	***String***	*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

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

- *Parameter **follow** indicates that several domain servers are working together (balanced).*
- *Users defined in any Active Directory node will be able to log in because DC=company,DC=com is defined as the base filter.*
- *Any user authenticated in Active Directory can log in because it does not have any filtering clause.*
- *Groups read by OpenKM can be defined in any Active Directory node because DC=company,DC=com is defined as the base filter.*

*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 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.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 showing login based on filtering users by roles

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***
- *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***

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

- *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 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 **logging in to OpenKM using 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 the next parameters are successfully configured, these lists will remain 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 common LDAP attribute for email is **mail**) **principal.ldap.mail.attribute=userPrincipalName**.*

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

Field / Property	*Type*	*Description*
principal.adapter	***String***	*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*
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* ***the user account filter is applied.***
- *Groups read by OpenKM can be created in any Active Directory node because DC=company,DC=com is defined as the base filter.*

Apply changes in the ***openkm.properties*** *file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable LDAP login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
ldap.server=ldap://192.168.0.13
ldap.manager.distinguished.name=CN=Administrator,CN=Users,DC=openkm,DC=local
ldap.manager.password=*secret*
ldap.base=DC=openkm,DC=local
ldap.role.attribute=cn
ldap.user.search.filter=(&(sAMAccountName={0})(|(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 for configuring 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.

```
<?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

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 the 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 logging 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.*

- *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 in 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***	*com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*cn=users,dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*
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 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 openkm.xml file; 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

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

Field / Property	*Type*	*Description*
principal.adapter	***String***	*com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.0.6:389*
principal.ldap.security.principal	***String***	*CN=Administrator,cn=users,dc=company,dc=local*
principal.ldap.security.credentials	***String***	*password*
principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*dc=company,dc=local*
principal.ldap.user.search.filter	***String***	*(objectclass=person)*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*dc=company,dc=local*
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}))*

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

principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*dc=company,dc=local*
principal.ldap.role.search.filter	***String***	*(objectclass=group)*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
principal.ldap.roles.by.user.search.base	***String***	*DC=company,DC=com*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0}))*
principal.ldap.users.by.role.attribute	***String***	*member*
principal.ldap.users.by.role.search.base	***String***	*dc=company,dc=local*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectClass=group)(cn={0}))*

Step 2 - configure login

*Apply changes in the **openkm.properties** file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable LDAP login.*

```
#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.

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

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://192.168.0.6:389/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>
```

Active Directory mixed configuration

The objective of this kind of configuration is to authenticate against Active Directory while obtaining 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 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 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 resumes, and you can continue working in the 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**. The adapter configuration parameter value is **"com.openkm.plugin.principal.MixedPrincipalAdapter"** as shown in the table below.*

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

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

- *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***	*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}))*

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

Step 1 - configure login

> *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 [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.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: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 synced

The objective of this kind of configuration is to authenticate against Active Directory and to 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) in the OpenKM database and removes any database user who 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 login; this configuration always works in real time.

Step 1 - configuration parameters

*We suggest l**ogging into OpenKM using 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 Administration. Until you successfully configure the following parameters, 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 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.*

- *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***	*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*
principal.ldap.mail.search.base	***String***	*dc=company,dc=local*

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

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.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: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.

LDAP best practices for filtering users and roles

Usually you only want to retrieve a subset of users and roles present in your LDAP, to be shown in user interface lists or be able to 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 user interface:

Field / Property	*Type*	*Description*
principal.ldap.user.search.filter	***String***	*(&(objectclass=person)(memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.role.search.filter	***String***	*(&(objectclass=group)(memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.users.by.role.search.filter	***String***	*(&(objectclass=group)(cn={0})(memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*
principal.ldap.roles.by.user.search.filter	***String***	*(&(objectClass=person)(sAMAccountName={0})(memberOf=CN=OpenKM,CN=users,dc=company,dc=com))*

For login restriction:

Modify the openkm.properties file as follows:

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

If you use openkm.xml, use:

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

Example based on 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 with the OpenKM role 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 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 restriction:

Modify the openkm.properties file as follows:

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

If you use openkm.xml, use:

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

*There are some characters 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.*

ClearOS LDAP configuration

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
```

*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 **logging in 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 the next parameters are successfully configured, these lists will remain empty.*

Go to Administration > Configuration parameters

Field / Property	*Type*	*Description*
principal.adapter	***String***	*com.openkm.plugin.principal.LdapPrincipalAdapter*
principal.database.filter.inactive.users	***Boolean***	*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***	
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*

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

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*
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. Easy way is to only specify the common name, and when signing use a policy that is permissive.*
>
> *You can create an alias (CNAME) called ldap if you wish.*

```
$ openssl req -out ldap.csr -key private/sys-0-key.pem -new

$ openssl ca -policy policy_anything -days 3650 -out /etc/pki/CA/certs/ldap.mydomain.local.crt -infiles ldap.csr
```

Add the CA cert to the nssdb:

```
$ certutil -A -d /etc/pki/nssdb/ -n "CA certificate" -t "CT,," -a -i /etc/pki/CA/ca-cert.pem
```

The LDAP server certificate needs to be saved in the nssdb with the name of Server-Cert.

```
$ openssl pkcs12 -export -inkey /etc/pki/CA/private/sys-0-key.pem -in /etc/pki/CA/certs/ldap.mydomain.local.crt -out

/root/Server-Cert.p12 -nodes -name 'Server-Cert'

$ pk12util -i /root/Server-Cert.p12 -d /etc/pki/nssdb/
```

Edit the file /etc/openldap/slapd.conf:

```
$ vim /etc/openldap/slapd.conf
```

```
TLSCACertificatePath    /etc/pki/nssdb
TLSCertificateFile      Server-Cert
TLSVerifyClient         never
```

Allow LDAP in ClearOS to be queried by OpenKM - edit the end of slapd.conf thusly:

```
access to *
    by self write
    by peername.ip=127.0.0.1 read
    ...
    by peername.ip=<OpenKM's IP address> read
```

Copy ClearOS' public CA certificate to the OpenKM server and add it to OpenKM's keystore:

```
$ keytool -import -trustcacerts -keystore /opt/openkm-6.2.3-community/java/jre/lib/security/cacerts -alias myca -file

/etc/pki/tls/certs/myca.pem
```

Step 3 - configure login

*Apply changes in the **openkm.properties** file.*

*The parameter "**okm.authentication.database**" is used to disable database login.*

*The parameter "**okm.authentication.ldap**" is used to enable ldap login.*

```
#Authentication
okm.authentication.database=false
okm.authentication.ldap=true

#LDAP
```

```
ldap.server=ldaps://ldap.mydomain.local
ldap.manager.distinguished.name=cn=Manager,ou=Internal,dc=mydomain,dc=local
ldap.manager.password=*secret*
ldap.base=dc=mydomain,dc=local
ldap.role.attribute=cn
ldap.user.search.filter=uid={0}
ldap.role.search.filter=member={0}
```

After 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 configuration in the openkm.xml file; 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.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=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.DefaultLd
 <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>

  <!--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>
```

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

LDAP troubleshooting

How to perform a nested group membership filter in Active Directory

The sample below demonstrates how to configure AD search to retrieve users who are members of a 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 the 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:

** 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 to use 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 not able to login

It may be a problem with LDAP DNS name resolution.

Solution:

To prevent this issueâ€”especially on Windows systemsâ€”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 domain server 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, login continues to take a long time, check with your IT security team to see if they have enabled micro-segmentation policies. This may be another reason why login takes so long.

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

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

Force all users to be members of ROLE_USER.

In large AD user repositories, it is not practical for the AD administrator to add a new group to all users; in this case, we can force via the XML login configuration that any logged-in user be a member of ROLE_USER. You must 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 (with SSL encryption) and you have not configured the trust certificate in the Java keystore.

*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**. 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 (version tested https://indy.fulgan.com/SSL/)*

The output will contain several entries delimited with the following:

```
-----BEGIN CERTIFICATE-----
aklfhskfadljasdl1340234234ASDSDFSDFSDFSDFSD
```

```
....
-----END CERTIFICATE-----
```

To prevent the loss of JKS after the Java upgrade, do not create the JKS under the JDK folders.

In the sample below, it is 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

The log may show a stack trace 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 the JVM configuration parameters in setenv.sh or setenv.bat by adding the following 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 with INFO level are shown, but you can increase the amount of logged information by raising the log level to WARN or DEBUG. The log configuration file is ***$TOMCAT_HOME/logback.xml****. Here you can change the error level 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 and 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 results are 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 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***	*Analyze 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 does not have read access for the user who ran the query. It should be used in combination with **am_more**.*

Application by default is configured with:

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

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 this user. When the user performs a search, OpenKM will return these documents as search*

results. However, the user cannot 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. However, these documents can be accessed using the API. You need to revoke permissions recursively to prevent the user from seeing these documents, so they will not 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 will not 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, 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 user view.*

Values:

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

For example, to enable download and property groups, you should set the mask to 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 can become a problem when every change affects several hundred (or even thousands) of nodes. There is an alternative security configuration parameter named **security.mode.multiple** which enables multiple changes at a time, making it faster.*

Field / Property	Type	Description
security.mode.multiple	**Boolean**	true

Some security changes can take a long time to be completed because they affect a large number of nodes. It's good practice to perform them as background tasks. For this, there is a configuration parameter named ***security.live.change.node.limit*** *used to set the limit on 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.

SSO Configuration

This section is deprecated. Any authentication configuration which is not managed by LDAP or database will be implemented as a custom external authentication.

***Before starting the SSO configuration you should have AD properly configured*.**

Take a look at LDAP configuration.

The aim of this configuration is to allow users to log in to OpenKM with their Windows credentials. To do it, it is necessary to configure some things, including client browsers.

In our case the whole system is integrated by the following components:

- *Active Directory on Windows Server 2008 and Windows Server 2012*
- *Tomcat server on a Linux machine*
- *Client on Windows 7 (browsers: Internet Explorer and Firefox)*

The following guides have been used as a starting point:

- *Spring Security, Single Sign-On bajo plataformas Windows con Active Directory y Kerberos V5*
- *LDAP configuration parameters*
- *http://projects.spring.io/spring-security-kerberos/*
- *http://docs.spring.io/spring-security-kerberos/docs/1.0.0.RC2/reference/htmlsingle/*

Active Directory

It is assumed that there is an Active Directory server installed and running (LDAP implementation) where all the user accounts that will be used later are going to be configured.

The steps to follow are:

1. *Create a Computer object inside Active Directory that will correspond to the Linux server*
    - *Once done, modify its properties and activate the following option:*
        - ***Trusted for delegation**: This step is important because, if not enabled, negotiation between clients and servers won't be possible.*
2. *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. **Create a domain user**. **This user must be used only and exclusively for this purpose because if this user is used to log in to another computer, it will not be able to be used to log on to the application server**. Once the user is created, modify its properties to prevent the password from being changed and to prevent the account from expiring.*

3. ***Generate 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:*

```
C:\>  ktpass -princ HTTP/tomcat.example.com@EXAMPLE.COM -mapuser user@EXAMAPLE.COM -crypto all -kvno 0 -ptype KRB5_NT_PRINCIPAL -pass * -out web.keytab
```

   It is important to highlight:

   - 1. ***tomcat.example.com@EXAMPLE.COM***: *Full name of the Tomcat server and its domain.*
     2. ***user@EXAMPLE.COM***: *The user created previously.*
     3. ***web.keytab***: *The file to which the private key that will be used by the Tomcat server is exported.*

4. *Register new SPN services. A Service Principal Name (SPN) needs to be set up with HTTP and a server name, for example: tomcat.example.org where Tomcat is running. This is used with a domain user, and its keytab is then used as a service credential:*

```
C:\> setspn -A HTTP/tomcat.example.org@EXAMPLE.ORG user
C:\> setspn -A HTTP/tomcat.example.org user
```

   Where tomcat.example.org is the machine where the Tomcat servlet container is running, and 'user' is the user previously created.

   Usually the previous command "ktpass" should create the service HTTP/tomcat.example.org. If it's already created, an error will be raised indicating it is already present.

5. *Verify the services have been correctly created and linked to the user:*

```
C:\> setspn -l user
```

6. *Finally create the groups that will be assigned to the users and from which the corresponding roles will be obtained (you should have them from the previous AD integration step). It is recommended to create:*

   - *An Organization Unit (OU) with name OpenKM*
   - *Inside the previous Organization Unit , create the groups that will be used as roles*
     - *ROLE_USER*
     - *ROLE_ADMIN*

If you want to create the SPN services again, the best way is to delete the user and create it again in the domain. Also the following command is available:

```
$ setspn -D openkmsso
```

But we do not recommend it. It's always better to delete the user.

Linux Machine

The first thing to do is to include the application server in the domain if it was not done previously. To do so, follow the guide:

http://cerowarnings.blogspot.com.es/2011/11/how-to-linux-en-dominio-windows.html

One thing to consider 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
```

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, the domain, KDC server, and allowed algorithms must be specified. The most important algorithms are ***arcfour-hmac-md5*** *and* ***rc4-hmac*** *which Microsoft Active Directory supports.*

Preserve almost all 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
```

Configuration files

The appContext.xml file might be modified and the ***security:http*** *entry may not be exactly as shown below (some additional attributes).*

To change the authentication type to use, it is necessary to modify the following two files:

- *$CATALINA_HOME/webapps/OpenKM/WEB-INF/appContext.xml:*

  Change the following line:

```
<security:http access-decision-manager-ref="accessDecisionManager" access-denied-page="/unauthorized.jsp">
```

to

```
<security:http access-decision-manager-ref="accessDecisionManager" access-denied-page="/unauthorized.jsp" en
```

Under this line add:

```
<security:custom-filter ref="spnegoAuthenticationProcessingFilter" position="BASIC_AUTH_FILTER" />
```

A detail to take into account in this file is that if it is required to filter access not only by authentication but also by roles, you need to replace the attribute IS_AUTHENTICATED_FULLY with an authorized role list separated by commas (ROLE_USER, ROLE_ADMIN)

- *$CATALINA_HOME/openkm.xml*

  Comment the line:

```
<security:authentication-manager alias="authenticationManager">
  <security:authentication-provider>
    <security:password-encoder hash="md5"/>
    <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>
```

And add the following:

```
<beans:bean id="spnegoEntryPoint" class="org.springframework.security.kerberos.web.authentication.SpnegoEntry
   <beans:constructor-arg value="/login" />
</beans:bean>

<beans:bean id="kerberosAuthenticationProvider" class="org.springframework.security.kerberos.authentication.Ker
   <beans:property name="userDetailsService" ref="ldapUserDetailsService" />
   <beans:property name="kerberosClient">
      <beans:bean class="org.springframework.security.kerberos.authentication.sun.SunJaasKerberosClient">
         <beans:property name="debug" value="true"/>
      </beans:bean>
   </beans:property>
</beans:bean>

<!--beans:bean id="kerberosServiceAuthenticationProvider" class="org.springframework.security.kerberos.authentic
<beans:bean id="kerberosServiceAuthenticationProvider" class="com.openkm.spring.OKMKerberosServiceAuthent
   <beans:property name="ticketValidator">
      <beans:bean class="org.springframework.security.kerberos.authentication.sun.SunJaasKerberosTicketValidato
```

```
            <beans:property name="servicePrincipal" value="HTTP/tomcat.example.com@EXAMPLE.COM" />
            <beans:property name="keyTabLocation" value="file:/path/to/web.keytab" />
            <beans:property name="debug" value="true" />
        </beans:bean>
    </beans:property>
    <beans:property name="userDetailsService" ref="ldapUserDetailsService" />
</beans:bean>

<!-- Ticket security manager -->
<security:authentication-manager id="authenticationManagerSpnego">
    <security:authentication-provider ref="kerberosAuthenticationProvider" />
    <security:authentication-provider ref="kerberosServiceAuthenticationProvider" />
</security:authentication-manager>

<beans:bean id="spnegoAuthenticationProcessingFilter" class="org.springframework.security.kerberos.web.authent
    <beans:property name="authenticationManager" ref="authenticationManagerSpnego" />
</beans:bean>

<beans:bean id="authorizationContextSource" class="org.springframework.security.kerberos.client.KerberosLdapCo
    <beans:constructor-arg value="ldap://server.example.com:389/dc=example,dc=com" />
    <beans:property name="loginConfig" ref="LoginConfig"/>
</beans:bean>

<beans:bean id="LoginConfig" class="org.springframework.security.kerberos.client.config.SunJaasKrb5LoginConfig
    <beans:property name="servicePrincipal" value="HTTP/tomcat.example.com@EXAMPLE.COM"/>
    <beans:property name="keyTabLocation" value="file:/path/to/web.keytab" />
    <beans:property name="useTicketCache" value="false" />
    <beans:property name="isInitiator" value="true" />
    <beans:property name="debug" value="true" />
</beans:bean>

<beans:bean id="ldapUserDetailsService" class="org.springframework.security.ldap.userdetails.LdapUserDetailsSe
    <beans:constructor-arg ref="FilterBasedLdapUserSearch" />
    <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator" />
    <beans:property name="UserDetailsMapper" ref="UserDetailsMapper" />
</beans:bean>

<beans:bean id="FilterBasedLdapUserSearch" class="org.springframework.security.ldap.search.FilterBasedLdapUs
    <beans:constructor-arg value="" />
    <!--beans:constructor-arg value="dc=example,dc=com" /-->
    <beans:constructor-arg value="(| (userPrincipalName={0}) (sAMAccountName={0}))" />
    <beans:constructor-arg ref="authorizationContextSource" />
</beans:bean>

<beans:bean id="defaultLdapAuthoritiesPopulator" class="com.openkm.spring.OKMLdapAuthoritiesPopulator"/>
<beans:bean id="UserDetailsMapper" class="org.springframework.security.ldap.userdetails.LdapUserDetailsMappe

<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="authorizationContextSource"/>
            <beans:property name="userSearch" ref="FilterBasedLdapUserSearch"/>
        </beans:bean>
    </beans:constructor-arg>
    <beans:constructor-arg ref="defaultLdapAuthoritiesPopulator" />
</beans:bean>

<!-- Alternative Ldap security manager -->
<security:authentication-manager alias="authenticationManager">
    <security:authentication-provider ref="ldapAuthProvider" />
</security:authentication-manager>
```

The important configuration data to take into account is:

- *ldap://server.example.com:389/dc=example,dc=com which is the location of the LDAP server.*
- *servicePrincipal, which is the service previously created on the server: HTTP/tomcat.example.org@EXAMPLE.COM*
- *keyTabLocation: that is the location in the file system of the Linux machine of the keytab file created on the Active Directory server. (It is possible to use classpath: but it is preferable to use file:)*

Configure LDAP in OpenKM

Once the previous configurations are done, the Tomcat server can be started up. To allow user names and roles to be available inside OpenKM, it is necessary to configure LDAP. To do this, it's recommended to follow the steps of LDAP configuration parameters.

Client browsers:

To allow automatic credential negotiation, it is also necessary to configure the client web browsers. This is detailed at:

http://docs.spring.io/spring-security-kerberos/docs/1.0.0.RC2/reference/htmlsingle/#browserspnegoconfig

- *Firefox*
  1. *Open Firefox.*
  2. *At address field, type **about:config**.*
  3. *In filter/search, type **negotiate**.*
  4. *Parameter **network.negotiate-auth.trusted-uris** may be set to default https:// which doesn't work for you. Replace this with your server address (tomcat.example.com in our example).*
- *Internet Explorer:*
  1. *Open Internet Explorer.*
  2. *Click Tools > Internet Options > Security tab.*
  3. *In Local intranet section, make sure your server is trusted, e.g. by adding it to a list (tomcat.example.com in our example).*
  4. *Ensure the "Enable Integrated Windows Authentication" option is enabled*
  5. *Add your server address (tomcat.example.com in our example)*
- *Chrome:*
  1. *Open Chrome.*
  2. *Click Configuration> Show Advanced Configuration > Change proxy configuration > Security tab.*
  3. *In Local intranet section, make sure your server is trusted, e.g. by adding it to a list (tomcat.example.com in our example).*
  4. *Ensure the "Enable Integrated Windows Authentication" option is enabled.*
  5. *Add your server address (tomcat.example.com in our example)*

Troubleshooting

SSO configuration is quite complex. Below you will find some clues and tools that can be useful while trying to configure it.

Enable log for SSO authentication

Modify the log configuration file and add the line:

```
<logger name="org.springframework.security" level="DEBUG" />
```

More information about log configuration can be found at Log configuration.

Check Kerberos authentication from Linux command line

Execute the command (it's case-sensitive and uppercase is important):

```
$ kinit user@EXAMPLE.ORG
```

If all is correct, you will be prompted for the password and the login from Linux will succeed. If it goes wrong, there is something wrong in your Kerberos Linux client configuration (krb5.conf).

Listing keys

Almost all problems we have found are caused by a badly created web.keytab file. If you get errors like:

```
rg.springframework.ldap.AuthenticationException: GSSAPI; nested exception is javax.naming.AuthenticationExcept
 at org.springframework.ldap.support.LdapUtils.convertLdapException(LdapUtils.java:182)
 at org.springframework.ldap.core.support.AbstractContextSource.createContext(AbstractContextSource.java:285)
 at org.springframework.ldap.core.support.AbstractContextSource.doGetContext(AbstractContextSource.java:119)
 at org.springframework.ldap.core.support.AbstractContextSource.getReadOnlyContext(AbstractContextSource.jav
 at org.springframework.ldap.core.LdapTemplate.executeReadOnly(LdapTemplate.java:791)
 at org.springframework.security.ldap.SpringSecurityLdapTemplate.searchForSingleEntry(SpringSecurityLdapTemp
 at org.springframework.security.ldap.search.FilterBasedLdapUserSearch.searchForUser(FilterBasedLdapUserSea
 at org.springframework.security.ldap.userdetails.LdapUserDetailsService.loadUserByUsername(LdapUserDetailsS
 at com.openkm.spring.OKMKerberosServiceAuthenticationProvider.authenticate(OKMKerberosServiceAuthenticati
 at org.springframework.security.authentication.ProviderManager.authenticate(ProviderManager.java:156)
 at org.springframework.security.kerberos.web.authentication.SpnegoAuthenticationProcessingFilter.doFilter(Spneg
 at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:342)
 at org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter.doFilter(AbstractAuthen
 at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:342)
 at org.springframework.security.web.authentication.logout.LogoutFilter.doFilter(LogoutFilter.java:110)
 at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:342)
 at org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter.doFilterInternal(We
 at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:107)
 at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:342)
 at org.springframework.security.web.context.SecurityContextPersistenceFilter.doFilter(SecurityContextPersistenceF
 at org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:342)
 at org.springframework.security.web.FilterChainProxy.doFilterInternal(FilterChainProxy.java:192)
 at org.springframework.security.web.FilterChainProxy.doFilter(FilterChainProxy.java:160)
 at org.springframework.web.filter.DelegatingFilterProxy.invokeDelegate(DelegatingFilterProxy.java:343)
 at org.springframework.web.filter.DelegatingFilterProxy.doFilter(DelegatingFilterProxy.java:260)
```

```
 at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243)
 at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210)
 at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:225)
 at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:169)
 at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:472)
 at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:168)
 at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:98)
 at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:927)
 at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:118)
 at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:407)
 at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:999)
 at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:565)
 at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:309)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
 at java.lang.Thread.run(Thread.java:745)
Caused by: javax.naming.AuthenticationException: GSSAPI [Root exception is javax.security.sasl.SaslException: G
 at com.sun.jndi.ldap.sasl.LdapSasl.saslBind(LdapSasl.java:168)
 at com.sun.jndi.ldap.LdapClient.authenticate(LdapClient.java:235)
 at com.sun.jndi.ldap.LdapCtx.connect(LdapCtx.java:2740)
 at com.sun.jndi.ldap.LdapCtx.<init>(LdapCtx.java:316)
 at com.sun.jndi.ldap.LdapCtxFactory.getUsingURL(LdapCtxFactory.java:193)
 at com.sun.jndi.ldap.LdapCtxFactory.getUsingURLs(LdapCtxFactory.java:211)
 at com.sun.jndi.ldap.LdapCtxFactory.getLdapCtxInstance(LdapCtxFactory.java:154)
 at com.sun.jndi.ldap.LdapCtxFactory.getInitialContext(LdapCtxFactory.java:84)
 at javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:684)
 at javax.naming.InitialContext.getDefaultInitCtx(InitialContext.java:307)
 at javax.naming.InitialContext.init(InitialContext.java:242)
 at javax.naming.ldap.InitialLdapContext.<init>(InitialLdapContext.java:153)
 at org.springframework.ldap.core.support.LdapContextSource.getDirContextInstance(LdapContextSource.java:43)
 at org.springframework.security.kerberos.client.KerberosLdapContextSource.access$001(KerberosLdapContextSo
 at org.springframework.security.kerberos.client.KerberosLdapContextSource$1.run(KerberosLdapContextSource.ja
 at org.springframework.security.kerberos.client.KerberosLdapContextSource$1.run(KerberosLdapContextSource.ja
 at java.security.AccessController.doPrivileged(Native Method)
 at javax.security.auth.Subject.doAs(Subject.java:356)
 at org.springframework.security.kerberos.client.KerberosLdapContextSource.getDirContextInstance(KerberosLdap
 at org.springframework.ldap.core.support.AbstractContextSource.createContext(AbstractContextSource.java:273)
 ... 39 more
Caused by: javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials pr
 at com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:212)
 at com.sun.jndi.ldap.sasl.LdapSasl.saslBind(LdapSasl.java:123)
 ... 58 more
Caused by: GSSException: No valid credentials provided (Mechanism level: Server not found in Kerberos database
 at sun.security.jgss.krb5.Krb5Context.initSecContext(Krb5Context.java:710)
 at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:248)
 at sun.security.jgss.GSSContextImpl.initSecContext(GSSContextImpl.java:179)
 at com.sun.security.sasl.gsskerb.GssKrb5Client.evaluateChallenge(GssKrb5Client.java:193)
 ... 59 more
Caused by: KrbException: Server not found in Kerberos database (7)
 at sun.security.krb5.KrbTgsRep.<init>(KrbTgsRep.java:70)
 at sun.security.krb5.KrbTgsReq.getReply(KrbTgsReq.java:192)
 at sun.security.krb5.KrbTgsReq.sendAndGetCreds(KrbTgsReq.java:203)
 at sun.security.krb5.internal.CredentialsUtil.serviceCreds(CredentialsUtil.java:309)
 at sun.security.krb5.internal.CredentialsUtil.acquireServiceCreds(CredentialsUtil.java:115)
 at sun.security.krb5.Credentials.acquireServiceCreds(Credentials.java:456)
 at sun.security.jgss.krb5.Krb5Context.initSecContext(Krb5Context.java:641)
 ... 62 more
Caused by: KrbException: Identifier doesn't match expected value (906)
 at sun.security.krb5.internal.KDCRep.init(KDCRep.java:143)
 at sun.security.krb5.internal.TGSRep.init(TGSRep.java:66)
 at sun.security.krb5.internal.TGSRep.<init>(TGSRep.java:61)
 at sun.security.krb5.KrbTgsRep.<init>(KrbTgsRep.java:55)
 ... 68 more
```

Probably your web.keytab is wrong. ***Ensure you have executed the "ktpass" command before executing the "setspn"***

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

commands.

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

Configuring Two Factor Authentication

Two-factor authentication (2FA) provides extra security for the OpenKM login. In addition to the typical login and password, once the user is authenticated, they will be asked for an extra six-digit code generated in a mobile application called Google Authenticator.

Compliance

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

It has been tested with:

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

It should work with other applications, such as:

- *OTP Auth (iOS)*
- *Step Two (iOS)*
- *FreeOTP*
- *Twilio Authy*
- *Duo Mobile by 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

Each user can configure their own account to use 2FA. To do so, they have to go to ***Tools*** > ***Preferences*** > ***User Configuration****.*

If 2FA has been enabled by the administrator, at the bottom of the User Configuration dialog you will be able to see a new option ***2FA****. Depending on whether 2FA is enabled, you will see:*

- *Disable button: if 2FA was already 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 which you should scan with the Google Authenticator application.*
- *A text box where you have to enter the verification code generated by Google Authenticator (once 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. *Focus your mobile toward 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 into OpenKM, after your login and password are verified, a new screen will appear where you will be required to enter a code that must be obtained from the Google Authenticator application you previously configured.*

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

Cache configuration

*OpenKM uses several caches to enhance performance. Some of these caches are used directly 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 appropriate for every installation. The cache engine used is the very popular EHCache, which is widely used and highly 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 must duplicate the provided configuration file into **$TOMCAT_HOME/ehcache.xml** and make the appropriate 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. If set to true, timeouts are ignored and elements never expire.*

*So, in this case, the cache called "com.openkm.cache.ldapPrincipalAdapter.general" will have a maximum of 1000 elements in memory and, in case of overflow, will be written to disk. These elements will be removed from this cache if 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 these caches' status by going to **Administration** > **Utilities** > **Cache stats**. In this section 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 the anonymous user

Behind the anonymous user, there's a standard user named "public," who 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 as the user "**public**" in the standard UI to check which nodes and options the user has access to (**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.).*

***Only the share preview option will 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

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/.

It has been included in OpenKM since v7.1.21. To enable it, you need to include these two lines in the openkm.properties configuration file:

```
management.endpoint.prometheus.enabled=true
management.metrics.export.prometheus.enabled=true
```

Once modified, you need to restart OpenKM and the endpoint http://127.0.0.1:8080/openkm/actuator/prometheus will be available. This endpoint is protected and will only be accessible to users with the ROLE_ACTUATOR role.

Setup validation

In order to verify that access is valid, open a terminal and execute this command:

```
$ http --auth actuator:s3cr3t0 http://localhost:8080/openkm/actuator/prometheus
```

If the user 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:
```

```
  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: ['172.17.0.1:8080']
    basic_auth:
      username: 'actuator'
      password: 's3cr3t0'
```

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.yaml:/etc/prometheus/prometheus.yml
```

You can start it by running:

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

For more info, take a look at the [Prometheus documentation](#).

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

Not able to connect from another computer on the network

> *Ensure you have access to the server from the other computers.*
>
> *Ensure your server firewall allows remote connections to the destination port.*

*If you expose the Tomcat port 8080, ensure it is bound 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, Tomcat must be restarted for the changes to take effect.*

Error downloading document as PDF

This error is usually caused by a missing LibreOffice installation or by an incomplete package installation.

IMAP SSL certificate error

First, obtain the certificate with the openssl command, and then import the certificate using the following:

```
$ keytool -import -alias mail.yourcompany.com -keystore /etc/java-6-sun/security/cacerts -file imapd.pem
```

> *The application must be restarted to refresh the SSL certificates—more information is available at Connecting+to+SSL+services.*

The OpenKM server has no direct Internet connection.

*If you need a proxy to access the Internet, edit your "**/etc/bash.bashrc**" file as root and put these lines at the end of your "**/etc/bash.bashrc**" file:*

```
export http_proxy=http://username:password@proxyserver.net:port/
export ftp_proxy=http://username:password@proxyserver.net:port/
```

You can omit the username and password if your proxy server has no password.

Access denied when installing OpenKM as a service on Windows 7

*The command shell works slightly differently in Vista. By default, when you run cmd.exe, it does not grant administrator rights. To get around it, **create a shortcut to cmd.exe on the desktop, open the Properties > Compatibility tab, and check the box for Administrator rights**. Then you can perform administrative tasks on the command line.*

Problems with character encodings

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

*Check the $TOMCAT/bin/setenv.sh or $TOMCAT/bin/setenv.bat file. You should set the JVM parameter "**-Dfile.encoding=utf-8**".*

> *The application must be restarted.*

PermGen errors

> *As of Java 8, there are no **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 the bind DN has a size limit for each LDAP query.

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

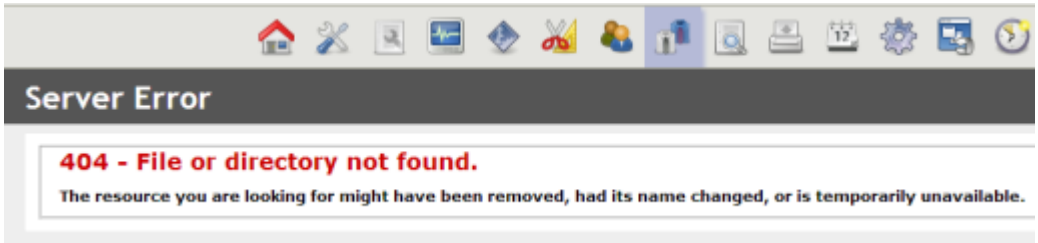
In OpenLDAP, the solution is to increase the bind DN limit; for example, change the size limit for the user used by the OpenKM LDAP connection.

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

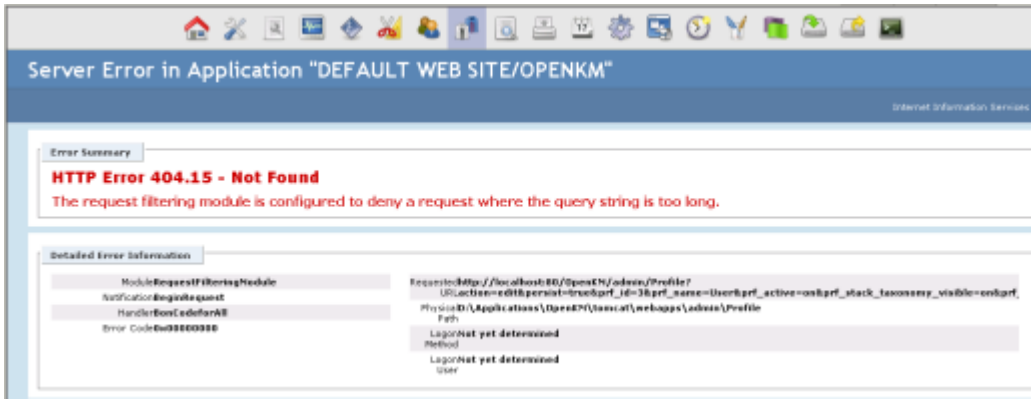
See also OpenLDAP Limits.

Behind the IIS proxy, you get error 404 or 404.15

By default, IIS, as a web server, has a limit for the query string sent to the server. If the limit is exceeded, you get an error screen on the client side:

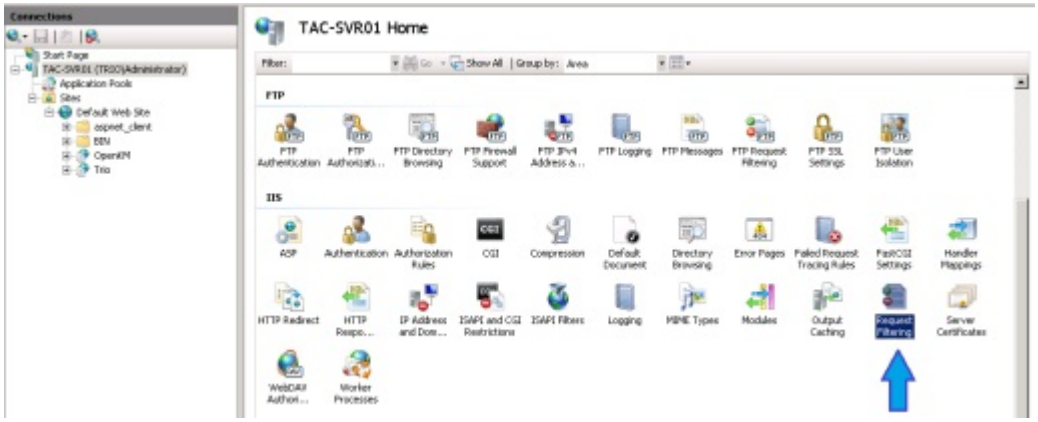


On the server side, we get a more detailed error:



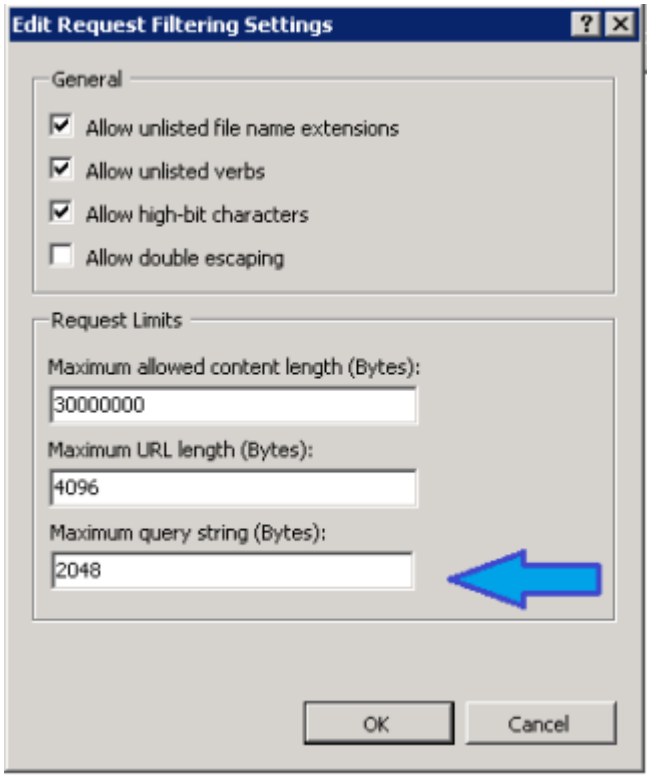
To solve this issue, we need to increase the maximum size of the query string on the web server:

- ***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:



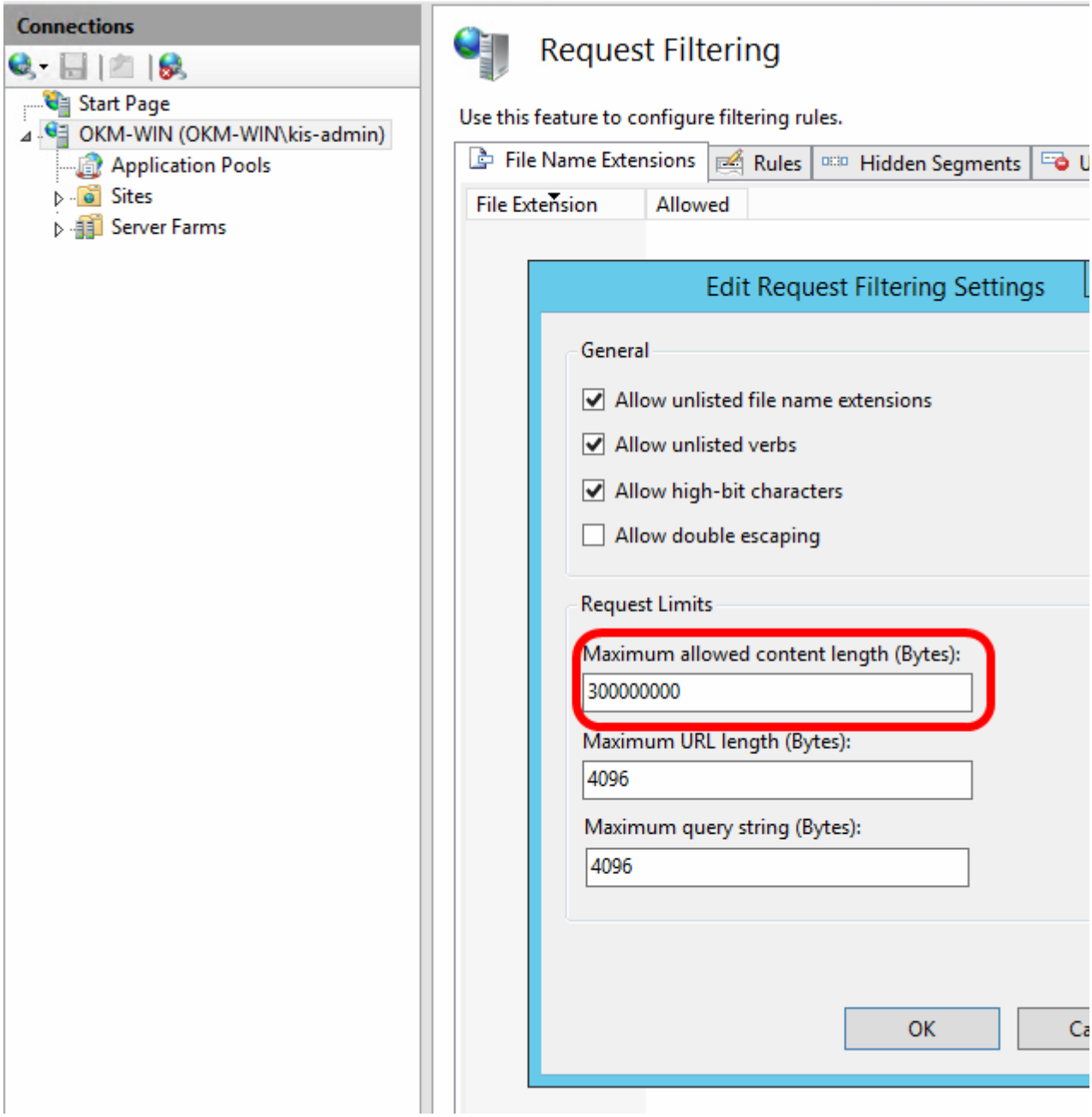
More information is available at the [OpenKM forum](#).

IIS gets IOException: Internal Server Error while uploading a 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*

- ***Edit** the **web.config** file and **add the following under <configuration> where "2097151"** is the maximum allowed in IIS:*

```
<system.web>
  <httpRuntime maxRequestLength="2097151" />
</system.web>
```

If there is no web.config, then create one from a text file and add the following:

```
<configuration>
  <system.web>
    <httpRuntime maxRequestLength="2097151" />
  </system.web>
</configuration>
```

WARN net.sf.ehcache.hibernate.strategy.AbstractReadWriteEhcacheAccessStrategy

If you see this message:

```
A soft-locked cache entry was expired by the underlying Ehcache.
If this happens regularly you should consider increasing the cache timeouts and/or capacity limit
```

Are you experiencing any performance or usability problems due to this warning? If not, then it seems like you should treat this warning as a warning.

*Otherwise, it looks like the solution is to increase the size of **maxElementsInMemory** to account for the difference in what is cached versus what is being used.*

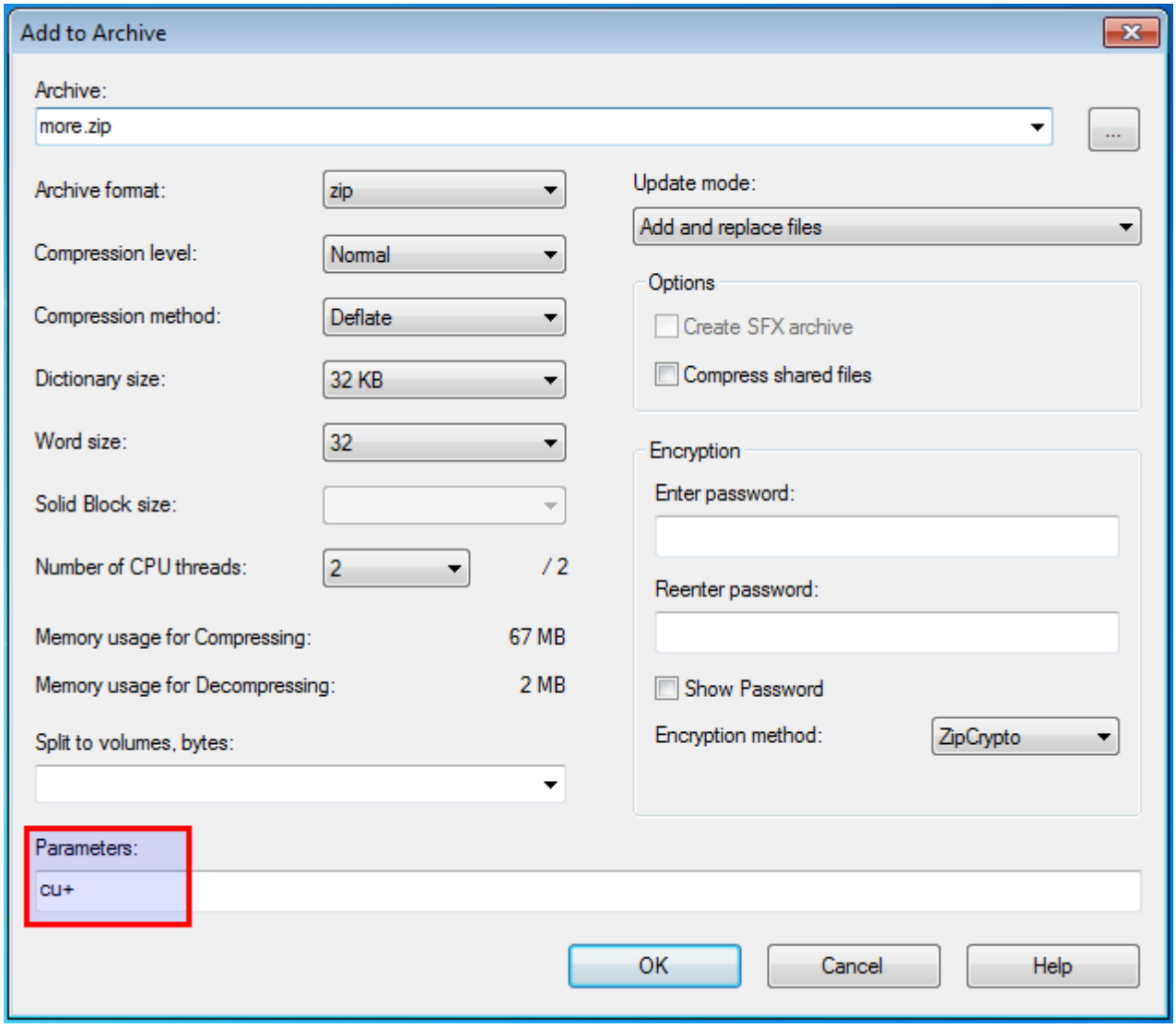
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

Client configuration

ZIP filenames with wrong charset

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

The other problem you may have is when you want to import a ZIP into OpenKM. In this case you can also get 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 the following:

- *Open normal.dot*
- *Go to the Tools menu / 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 quickly help you in error identification and resolution, please try to follow the guidelines below:

- *Provide a **detailed description** of the issue; this will help us understand the problem you have.*
- *When possible **take screenshots of all the steps you followed**. The idea is to be able to reproduce the error or at least better understand the context where 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 the download link using online file sharing tools like:*
>
> - *https://transfer.sh/*
> - *https://filepush.co/*
> - *https://filetransfer.io/*
> - *https://justbeamit.com/*

Getting the stack trace

Linux

*The **openkm.log** file is in the **$TOMCAT/logs folder**. For example, in Linux a common location might be "**/home/openkm/tomcat-8.5.69/logs**".*

To watch how openkm.log changes in real time, you can use the command line:

```
/home/openkm/tomcat-8.5.69/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-8.5.69/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-8.5.69\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-8.5.69\extras***

Sample of stack trace capture

When you capture the stack trace, it is 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 - E
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
```

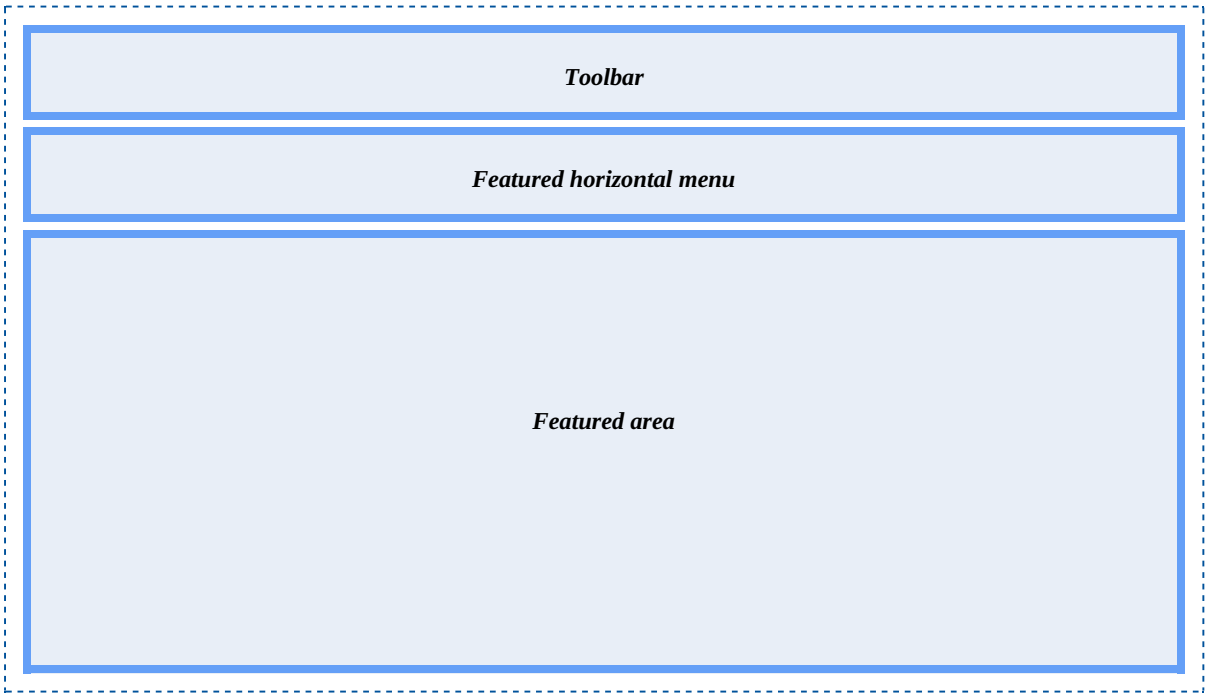
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

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. The following are shown:* • *Application version.* • *Installation UUID.*
	Configuration parameters	*Almost all configuration parameters are set here.* *Changes made here affect the entire application.*

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

	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 and roles.* *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, such as DOCUMENT_CREATION.*
		Adds or removes scheduled crontab tasks. The application uses these tasks to perform

	Crontab	*actions such as indexing documents and antivirus analysis.*
	Languages	*Add, modify, or remove languages and manage translations.*
	Utilities	*A collection of advanced utilities:* • *Admin permissions.* • *Cache Stats.* • *Check email.* • *Check text extraction.* • *Clean conversion cache.* • *Create a support zip.* • *Database import.* • *Database query.* • *Hibernate stats.* • *LDAP search.* • *List indexes.* • *LogCat.* • *Plugins* • *Profile stats.* • *Purge data.* • *Purge trash.* • *Rebuild indexes.* • *Repository checker.* • *Repository export.* • *Repository import.* • *Scripting.* • *System properties.*

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 as a new list element).*
- *Text.*
- *HTML (HTML tags allowed).*
- *File.*

To create your parameters, read Creating your own parameters.

Edit a parameter

- *Click the* ***edit icon****.*
- *View and* ***modify the value****.*
- *Click the* ***Edit button*** *to confirm the change.*

Delete a parameter

You should not delete parameters if you're not sure about the implications of this action.

- *Click the* ***remove icon*** .
- *In the* ***Delete confirmation view****, click the* ***Delete button****.*

Backup configuration parameters

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

- *Click the **Configuration parameters export icon** at the top right .*
- *The browser will download an SQL file with the configuration.*

New configuration parameter

- *Go to the upper-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, OpenOffice, 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, which cannot cover all cases, but is an excellent approximation to discard most problems.*

- *Click the **Check icon** at the top right .*
- *It will show the **Configuration check view**. If some parameters are misconfigured or wrong, they will be shown in red.*

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

Recommended configuration parameters

There are some recommended changes to 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 mail notification body.* *Message: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}*
notification.message.subject	***Text***	*This parameter is used in mail notifications. It helps users identify that these emails are coming from OpenKM.* *OpenKM - NOTIFICATION*
subscription.message.enabled	***Boolean***	*Determines whether the subscription is evaluated and the message is sent. It is enabled by default, 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 mail user subscription notification body.* *Date: ${eventDate?string("yyyy-MM-dd HH:mm:ss")}* *Document: ${documentPath}* *User: ${userId}* *Event: ${eventType}* *Comment: ${subscriptionComment}*
subscription.message.subject	***Text***	*This parameter is used in the mail subscription notification subject. 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***	*Enables or disables concurrent text extraction threads.*
text.extraction.batch	***Integer***	*It indicates the number of documents processed in each indexing cycle.* *When parameter "**text.extraction.concurrent" is enabled**, it is good practice to set a multiple of the **"text.extraction.threads"** value.*
text.extraction.threads	***Integer***	*The 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.*
mail.extraction.concurrent	***Boolean***	*Enables or disables concurrent mail extraction threads.*
mail.extraction.batch	***Integer***	*It indicates the number of emails processed in each indexing cycle.* *When parameter **"mail.extraction.concurrent" is enabled**, it is good practice to set a multiple of the **"mail.extraction.threads"** value.*
mail.extraction.threads	***Integer***	*The 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 the ImageMagick convert tool.*
system.ocr	***String***	*Path to the OCR text extractor engine.*
system.openoffice.program	***String***	*Path to the LibreOffice installation.*

logo.login	***File***	*Logo for the login page.* *It automatically tries to resize the image to 316px × 74px (suggested background #F1F3F5).*
logo.mobile	***File***	*Logo for the mobile login page.* *It automatically tries to resize the image to 161px × 38px (suggested background #F1F3F5).*
logo.report	***File***	*The default logo used in JasperReports.* *It automatically tries to resize the image to 150px × 35px (suggested background #FFF000).*
logo.favicon	***File***	*Small icon on the left side of the browser's address bar.* *The uploaded file must be a valid icon (16px x 16px).*

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

Preview parameters

*In this section you can find the most common configuration parameters related to preview. Some of these commands are included in OpenKM by default, like **pdf2swf** or **ffmpeg** (binaries 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 extractor tool.* `/usr/bin/pdfimages -j -f ${firstPage} -l ${lastPage} ${fileIn} ${imageRoot}`
system.openoffice.program	***String***	*Path to OpenOffice or LibreOffice executable. Used in* ***OfficeDirectToPdfConversion****. If you are using this converter, LibreOffice is not required to start as a service so you can avoid it by setting the* ***system.openoffice.path*** *configuration property to an empty value and set* ***system.openoffice.server*** *to "local".* `/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 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 extractor tool.* *C:\openkm\tools\pdfimages -j -f ${firstPage} -l ${lastPage} ${fileIn} ${imageRoot}*
system.openoffice.program	***String***	*Path to OpenOffice or LibreOffice executable. Used in **OfficeDirectToPdfConversion**. If you are using this converter, LibreOffice is not required to start as a service so you can avoid it by setting the **system.openoffice.path** configuration property to an empty value and set **system.openoffice.server** to "local".* *C:\Program Files\LibreOffice\program\soffice.exe*
system.ghostscript	***String***	*Path to the Ghostscript conversion tool.* *C:\openkm\bin\gswin32c.exe*

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

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 the sAMAccountName attribute value {0} â€" this parameter indicates the userId â€" 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.**** *are used to retrieve the user list.*
- ***principal.ldap.username.**** *are used to retrieve the user name.*
- ***principal.ldap.mail.**** *are used to retrieve user mail addresses.*
- ***principal.ldap.role.**** *are used to retrieve the role list.*
- ***principal.ldap.roles.by.user.**** *are used to retrieve a user's roles.*
- ***principal.ldap.users.by.role.**** *are used to retrieve users of a role.*

When retrieving data, arguments are injected into the option filters. For example, to get the user mail, the application uses parameter {0} to set the argument value â€" userId â€" in the filter 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***	Class used to connect with LDAP.
system.login.lowercase	***String***	By default it is empty. When connecting to AD (Microsoft Active Directory) it 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.
principal.ldap.referral	***String***	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 in the form of 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: "***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 set to "**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 base nodes.*
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 base nodes.*
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***	*com.openkm.plugin.principal.LdapPrincipalAdapter*
system.login.lowercase	***String***	*true*
principal.ldap.server	***String***	*ldap://192.168.1.20:389*
principal.ldap.security.principal	***String***	*CN=Administrator,OU=OpenKM,DC=company,DC=com*
principal.ldap.security.credentials	***String***	*password*

principal.ldap.referral	***String***	
principal.ldap.users.from.roles	***Boolean***	*false*
principal.ldap.user.attribute	***String***	*sAMAccountName*
principal.ldap.user.search.base	***List***	*DC=company,DC=com*
principal.ldap.user.search.filter	***String***	*objectclass=person*
principal.ldap.username.attribute	***String***	*cn*
principal.ldap.username.search.base	***String***	*DC=company,DC=com*
principal.ldap.username.search.filter	***String***	*(sAMAccountName={0})*
principal.ldap.mail.attribute	***String***	*mail*
principal.ldap.mail.search.base	***String***	*DC=company,DC=com*
principal.ldap.mail.search.filter	***String***	*(sAMAccountName={0})*
principal.ldap.role.attribute	***String***	*cn*
principal.ldap.role.search.base	***List***	*DC=company,DC=com*
principal.ldap.role.search.filter	***String***	*objectclass=group*
principal.ldap.roles.by.user.attribute	***String***	*memberOf*
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.ui.context	***String***	*The context of the application.* */kcenter/*
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*
security.mode.multiple	***Boolean***	*Enables an alternative security mode that allows the application to apply several changes simultaneously, but only one change is used at a time. By default, the value is "false".* *After the parameter is changed, refresh the browser for the change to take effect.* *false*
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.*

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

		It automatically tries to resize the image to 150px × 35px (suggested background #FFF000).
text.banner	***String***	*Text is shown at the top of the logo on the landing page.* *It automatically tries to resize the image to 51px × 13px (suggested background #E5E5E1).*
thumbnail.size.lightbox	***String***	*Maximum size - width × height - of the lightbox thumbnail.* `640×640`
thumbnail.size.properties	***String***	*Maximum size - width × height - of the thumbnail in the properties view.* `280×280`
thumbnail.size.search	***String***	*Maximum size - width × height - of the thumbnail in search view.* `180×180`
download.zip.live.max.files	***Integer***	*Sets the maximum number of files allowed to be downloaded in live mode. Once this number is exceeded, the application will notify the user that the ZIP creation will be executed in background mode. After it is finished, it will notify the user by email with a download URL. The value**0** indicates an **unlimited number of files**.* *Downloading a few selected files and folders can take time. On the server side, the application creates a temporary folder where all data is packed as a ZIP file and then forwarded to the end-user interface.* `100`
download.zip.live.max.file.size	***String***	*Sets the maximum 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 is finished, it will notify the user via email with a download URL. A value of **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. On the server side, the application creates a temporary folder where all data is packed as a ZIP file and then forwarded to the end-user interface. *100MB*
min.search.characters	***Integer***	*Sets the minimum 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.

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 the 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 mail field va it will use the default From mail address for all the emails sent by the applicatior address).* *When the mail server restricts changing the default From mail field (the general@mail.com account to send mail with a From mail field v is usually set to prevent spam), then this parameter must be designat* *true*
notification.message.subject	***Text***	*The subject of the message is sent when a user has chosen to notify someone else.* *OpenKM - NOTIFICATION*
notification.message.body	***HTML***	*Body of the message sent when a user has chosen to notify someone else.* *ReplaceMessage: ${notificationMessage}* *User: ${userId}* *<#list documentList as doc>Document: ${doc.path}* *</#list>*
subscription.message.subject	***Text***	*The subject of the message is sent automatically in response to some event. This* *OpenKM - ${eventType} - ${documentPath}*
subscription.message.body	***HTML***	*The body of the message is sent automatically in response to some event. This ha* *Date: ${eventDate?string("yyyy-MM-dd HH:mm:ss")}* *Document: ${documentPath}* *User: ${userId}* *Event: ${eventType}* *Comment: ${subscriptionComment}*

download.message.body	***HTML***	*The body of the message is sent when a user shares a document with others for d* *Your archive is ready. Please download it from ${url}*
download.message.subject	***String***	*The subject of the message is sent when a user shares a document with others fo* *OpenKM - DOWNLOAD*
download.token.validity	***Integer***	*The number of days the download URL will be available.* *7*
template.message.body	***String***	*Set the basic template for all notifications.*

```
<div style="margin: 0; padding: 0; width: 100%; min-height: 100%;">
<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">
<div style="background: #fff;">
<table width="100%" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td>
<table width="100%" cellspacing="0" cellpadding="0">
<tbody>
<tr>
<td bgcolor="#018ca9" width="100%" height="6px"> </td>
</tr>
</tbody>
</table>
</td>
</tr>
<tr>
<td>
<div style="border-top: solid 1px #88888b; min-height: 5px; max-height
</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>
notification.url	***String***	*The notification URL parameter is the template URL that OpenKM uses for build emails, records).* *http://localhost:8080/openkm/kcenter/#*

Substitutions

*For **notification.message.body** property:*

- ***${documentUrl}** shows the node URL.*
- ***${documentPath}** shows the node path.*
- ***${documentName}** shows the node name.*
- ***${userName}** displays the user name who caused the event.*
- ***${userId}** shows the user ID who caused the action.*
- ***${notificationMessage}** shows the notification message.*

*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 Creating your own Version Number Adapter.*

Field / Property	*Type*	*Description*
version.numeration.adapter	***String***	*Numeration schema.* *Available schemas:* • *com.openkm.plugin.vernum.PlainVersionNumerationAdapt* • *com.openkm.plugin.vernum.PlainZeroVersionNumerationA* • *com.openkm.plugin.vernum.MajorMinorVersionNumeratio* • *com.openkm.plugin.vernum.BranchVersionNumerationAda* • *com.openkm.plugin.vernum.MajorMinorReleaseVersionNu* • *com.openkm.plugin.vernum.VersionLetterAdapter* *com.openkm.vernum.MajorMinorVersionNumerationAdapter*
version.numeration.format	***String***	*Numeration number format. Used in the* ***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 documents already exist 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 at the beginning of 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 documents already exist 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, the migration from ***PlainVersionNumerationAdapter*** *to* ***MajorMinorVersionNumerationAdapter****.*

```
-- HQL queries
UPDATE NodeDocumentVersion ndv SET ndv.name = '01' WHERE ndv.name='1.0';
UPDATE NodeDocumentVersion ndv SET ndv.name = '02' WHERE ndv.name='1.1';
UPDATE NodeDocumentVersion ndv SET ndv.name = '03' WHERE ndv.name='1.2';
```

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

Password validation configuration

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, the validator.password property must be changed to ***com.openkm.validator.password.CompletePasswordValidator*** *or Creating your own password validator.*

Field / Property	*Type*	*Description*
validator.password	***String***	*Description* *Available:* • *com.openkm.validator.password.NoPasswordValidator* • *com.openkm.validator.password.CompletePasswordValidator* *com.openkm.validator.password.NoPasswordValidator*
validator.password.min.length	***String***	*Minimum length allowed.* *Numeric value greater than 0.*
validator.password.max.length	***String***	*Maximum length allowed.* *Numeric value greater than 0.*
validator.password.min.lowercase	***String***	*Minimum number of lowercase characters.* *Numeric value greater than 0.*
validator.password.min.uppercase	***String***	*Minimum number of uppercase characters.* *Numeric value greater than 0.*
validator.password.min.digits	***String***	*Minimum number of digits.* *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.

		Numeric value greater than 0.

Sample configuration

- *Must have at least 8 characters.*
- *Must have at least 1 uppercase 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 entire repository to be read-only. This is useful when doing maintenance tasks, or during migration when you want to keep the system available to your users but 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 **a Lucene index rebuild**, the system will **switch automatically 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 super user ID. By default, the super user 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 the user 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, the landing page provides the option to reset their password and to be notified by mail with a new one. 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. The default value is "simple".* *Values:* • *simple.* • *recursive.* • *read_recursive.*

		simple *Take a look at Security configuration*
security.search.evaluation	***String***	*Sets the security search manager and determines how the 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, adding, removing, or modifying property groups (metadata), or compacting history. The default value is empty.* *Values:* • *DOWNLOAD = 1024;* • *COMPACT_HISTORY = 4096;* • *PROPERTY_GROUP = 8192;* *For example, to enable download and property groups, you should use the mask 9216 (1024+8192).* *9216* *Take a look at Security configuration*
security.live.change.node.limit	***Integer***	*When security changes affect more than the value set in this property, they are applied as a background task.* *100* *Take a look at Security configuration*
system.login.lowercase	***Boolean***	*By default it is empty. When connecting to AD (Microsoft Active Directory) it must be set to "**true**", which forces all users to log in with a lowercase user ID. The reason is that 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 the creation of user names and role names.* ***^[a-zA-Z0-9_]+$*** *Regular expression sample for Arabic:* ***^[\u0600-\u06FF]+$*** *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 (autocompletion allowed). This parameter allows disabling password autocompletion in the login frame. This is a security measure on untrusted clients.*
default.security.recursive.role	***String***	*Sets the role that is used to identify which users are able to set recursive security.*

		ROLE_SECURITY_RECURSIVE
security.login.failed.attempts	***Integer***	*The number of failed login attempts before the user is locked. A value of 0 indicates unlimited failure attempts, and the user will never be locked.* *0*

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 the 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 tab document extension

The External tab document feature shows extra tabs in tab document view based on an external URL request.

Field / Property	Type	Description
extension.external.tab.document	**List**	*The format for each line is **title;URL**, where the character ";" is used as the separator between the title of the tab and the URL of the contents.* `OpenKM;http://www.openkm.com`

External tab folder extension

The External Tab Folder feature helps you display the tab content in the properties tab based on an external URL request when a node folder is selected.

Field / Property	Type	Description

extension.external.tab.folder	**List**	*Each row is a 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`

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 node mail is selected.

Field / Property	*Type*	*Description*
extension.external.tab.mail	***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*

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.

Field / Property	*Type*	*Description*
extension.external.tab.record	***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*

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

Performance configuration parameters

Text extractor queue

There are several configuration parameters to adapt the document text extraction to your needs. Text extraction is one of the processes that can use a lot of hardware resources (memory and CPUs). An aggressive text extraction policy can decrease the performance of the application and affect the end user's feelings about the user interface.

When documents are uploaded, they are automatically added into ***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.

In case the application needs to index a lot 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 need 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 parameter "**text.extraction.concurrent**" is enabled, it is good practice for it to be 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 to be used for the text extraction feature.* *This number should be less than or equal to the number of hardware cores.*

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

		2

There's a crontab task named "**Mail extractor worker**" at **Administration** > **Crontab** that 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 parameter "**mail.extraction.concurrent**" is enabled, it is good practice for it to be a multiple of **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 to be used for the mail extraction feature. This number should be less than or equal to the number of hardware cores. 2

Download files and folders as ZIP file

It is not good practice to download large 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 finally be forwarded to the user interface. To prevent blocking the user interface while waiting for the process to complete, there are two parameters that will notify the user that 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 download in live mode.

download.zip.live.max.files	***Integer***	*100*
download.zip.live.max.file.size	***String***	*Set the maximum total 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 experience of the user interface. The antivirus checker can work in live mode (as the uploading process antivirus checks the document, and if a virus is detected, an immediate warning is raised and the document is not incorporated into the repository) or in background mode.

> *The antivirus checker in live mode can dramatically degrade the user interface and the end user's experience. Keep in mind that to analyze some documents, the antivirus can take 3 seconds or more to check them, which will be added to the upload processing time.*

> *In case the application needs to analyze a lot 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 takes control of the background antivirus checker.* *When the background check is disabled, the antivirus is processed while uploading any document. That can add extra time to the uploading process (in some cases, an 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 3x or more with this option disabled.*

		Our suggestion is to enable it. *true*
background.antivirus.check.threads	***Integer***	*The number of concurrent threads to be used for antivirus scanning.* *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 parameter "**background.antivirus.check**" is enabled, it is good practice for it to be 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 of rebuilding Lucene indexes: sequential or parallel. By default, sequential re-indexing is enabled, but you can select the mode using the "**hibernate.indexer.rebuild**" configuration parameter.*

Field / Property	*Type*	*Description*
hibernate.indexer.rebuild	***String***	*By default, set to "**flush**". Possible values are "**mass**", "**flush**", and "**delayed**". Take a look at the table below to see the different value details.*

*You can use the "**hibernate.indexer.batch.size.load.objects**" configuration parameter to indicate to Hibernate how many objects should be handled 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 reindex the repository in parallel, using several threads. This may be faster, but it is also problematic in terms of resources. The repository will be in read-only mode until completed.* *You have several configuration properties to tune its performance:*

	• ***hibernate.indexer.batch.size.load.objects***: *batch size used to load the entities.* *30* • ***hibernate.indexer.threads.load.objects***: *number of threads used to load the entities.* *4*
Flush to Indexes	*This mode will rebuild the index sequentially in batches. It's less resource hungry but may be slower. The repository will be in read-only mode until completed.* *You have several configuration properties to tune its performance:* • ***hibernate.indexer.batch.size.load.objects***: *batch size used to load the entities.* *30*
Delayed Indexer	*There's a cron task named "Lucene Delayed Indexer" which processes the queue of nodes to be indexed in the background.* *This queue is filled up when you go to Administration > Tools > Rebuild indexes and choose "Lucene indexes". When the delayed indexer is enabled, the nodes are included in the batch queue to be processed with this background process.* *This mode will rebuild the index sequentially in batches. It's less resource hungry 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 working.* *You have several configuration properties to tune its performance:* • ***hibernate.indexer.delayed.max.objects***: *how many nodes will be processed in a batch.* *1000* • ***hibernate.indexer.batch.size.load.objects***: *number of nodes to read from the database at the same time.* *4*

Execution timeout

The application executes external applications to process documents, for example, extracts text with an OCR engine, analyzes with antivirus software, transforms documents to other formats, among other actions. Sometimes these processes can take a

*long time or for some reason do not finish correctly, leaving processes 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 the activity of the users, but sometimes these actions don't need to be logged and fill your activity log table.

The table where the activity log is stored - OKM_ACTIVITY - may grow quickly, storing millions of records.

*There is a configuration property named "**activity.log.actions**" where you can set which actions to log. By default, this is set to the most common or interesting actions. You can use regular expressions to define these actions. Read Java Regex Tutorial for more info about Java regular expressions.*

Field / Property	Type	Description
activity.log.actions	**List**	*LOGIN* *LOGOUT* *CREATE_.** *DELETE_.** *PURGE_.** *MOVE_.** *COPY_.** *CHECKOUT_DOCUMENT* *CHECKIN_DOCUMENT* *GET_DOCUMENT_CONTENT.** *MISC_TEXT_EXTRACTION_FAILURE*

For a complete list of actions, see Activity log.

Dashboard activity

The OpenKM dashboard displays a global view of application activity. For this, the application needs to process a large amount of information, which can decrease the overall performance of the application (for example, to get the latest documents uploaded into the system, OpenKM might execute slow database queries). Therefore, we suggest that 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 this, 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.autoregister_listeners	***String***	*Enable or disable the Lucene search engine.* *Allowed values:* • *on* • *off* *on*
spring.jpa.properties.hibernate.search.default.worker.execution	***String***	*Set the Lucene worker execution mode. By default it is set to "sync", which means that an entity modification will not complete until the Lucene index has been properly updated. In the case of "async", these entity modifications will be faster because the Lucene work is added to a queue and processed in a background thread.* *Allowed values:* • *sync*

		• *async* *sync*

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

OpenOffice & LibreOffice configuration parameters

As part of the application startup, OpenKM executes an OpenOffice or LibreOffice service. That's used internally for conversion purposes, for example, to convert doc files to PDF.

Starting the OpenOffice or LibreOffice process before OpenKM starts up can cause the service that OpenKM executes internally during startup to fail.

The OpenOffice or LibreOffice service can use a lot of hardware resources (CPU up to 100%).

Field / Property	*Type*	*Description*
system.openoffice.program	***String***	*Path to OpenOffice or LibreOffice.*
system.openoffice.dictionary	***String***	*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 view results, user 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***	*Specifies whether empty lines must 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***	*Specifies 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 application's performance and negatively affect end users' experience of the user interface.

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

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

> ⚠ *The antivirus checker in live mode can dramatically decrease the end-user 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****.* ℹ *There's a* ***crontab*** *task (****Administration > Crontab****) named* ***Antivirus Checker Worker*** *that takes control of the background antivirus checker.* *For more information, see Built-in crontab tasks.* ⚠ *When background check is disabled, the antivirus is processed while uploading any document. That can add extra time to the upload process (in some cases more than 3 seconds 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 3 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 for the antivirus.* ℹ *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.

background.antivirus.check.batch	***Integer***	*Indicates the number of documents to be processed on each indexing cycle.* *When parameter "**background.antivirus.check is enabled**", it is good practice to be a multiple of the "**background.antivirus.check.threads**" value.*

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

Other configuration parameters

Property	*Type*	*Description*
spring.jpa.properties.hibernate.search.analyzer	***String***	*This parameter is used to change the default Apache Lucene Search Engine Analyzer.* *org.apache.lucene.analysis.standard.StandardAnalyze* *More information at Configuring Lucene Analyzer.* *This parameter must be set in the $TOMCAT_HOME/openkm.properties file.*
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.* *This parameter must be set in the $TOMCAT_HOME/openkm.properties file.*
system.ocr.rotate	***String***	*This parameter forces OCR to process rotated documents.* *The feature is useful, for example, in upside-down scanned pages, where standard OCR direction will fail.* *The parameter is a collection of degrees separated by the character ";"* *90;180;270;* *More information at Configuring OCR engine.*
system.ocr.spell.check	***Boolean***	*When it is enabled, the text extracted by OCR is validated by the spell checker.* *The application must have an enabled dictionary; see the parameter "**system.openoffice.dictionary**" at OpenOffice*

		LibreOffice configuration parameters.
system.document.name.mismatch.check	***Boolean***	*This parameter is used to enable or disable the name check when a new version of a document is uploaded.*
max.search.results	***Integer***	*This parameter sets the limits of the search results. For example, a value of 500 means you will never get more than 500 results for any query.* *500* *The reason why the number of results is limited is that some queries can retrieve a huge number of results, especially when the queries are done across API methods.*
system.keyword.lowercase	***Boolean***	*This parameter is used to force all keywords to lowercase.* *The application is case-sensitive, which means a user can set two keywords like "keyone" and "KeyOne" and, for the application, these two keywords are distinct. To prevent this problem, it is always a good idea to force all keywords to lowercase.*
tomcat.connector.uri.encoding	***String***	*By default, Tomcat Connector encoding is set to "**ISO-8859-1** but you can change it with this parameter.* *When you set the "**URIEncoding**" parameter in Tomcat to "**UTF-8**", you also need to set this parameter to "**UTF-8**".* *URIEncoding="UTF-8"* *ISO-8859-1*
mail.session.debug	***Boolean***	*Enable mail session debug. It's useful to detect problems when the application is not sending emails.*

mobile.theme	***String***	*There are five mobile themes ("a", "b", "c", "d", "e").* *More information about themes at jQuery mobile themes.* *b*
plugin.debug	***Boolean***	*When enabled, it shows debug information about the application plugin feature.*
principal.database.filter.inactive.users	***Boolean***	*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 character ";".* **~;*.bak*
restrict.file.mime	***Boolean***	*Restrict file name extensions based on existing ones at* ***Administration*** *>* *MIME types.*
restrict.duplicated.documents	***Boolean***	*It is not allowed to upload the same document more than once* *Internally, the verification is based on the document checksum value.*
native.sql.optimizations	***Boolean***	*For best performance, some database operations are done wit SQL rather than HQL.*
repository.content.checksum	***Boolean***	*When it is enabled, the document checksum is calculated for each uploaded document.*
system.pdf.force.ocr	***Boolean***	*Forces all PDF files to be processed with the OCR engine.*

system.catdoc.xls2csv	***String***	*Native XLS to CSV converter tool. Used in* ***NativeMsExcelTextExtractor****. In Debian and other Ubuntu-based distros it is included in the* ***catdoc*** *package.* */usr/bin/xls2csv ${fileIn}*
system.maintenance	***Boolean***	*When the application is enabled, it goes into "maintenance mode", which means nobody except the okmAdmin user will be able to log in.*
system.strings	***String***	*Configure the Strings tool.* */usr/bin/strings ${fileIn}*
system.zbar	***String***	*Configure the barcode extractor tool.* */usr/bin/zbarimg*
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 repository search.*
version.append.download	***Boolean***	*Append the document version to the document file name when downloading a document from OpenKM.*
webservices.visible.properties	***List***	*List of configuration properties that can be viewed by users without administrator grants via the webservices.*
rss.news	***Boolean***	*Enable or disable the news feature on the login page.*
rss.news.box.width	***Integer***	*The width of the news box on the login page.* *300*
rss.news.max.size	***Integer***	*The maximum number of news items retrieved from the news server.* *10*

		News is cached from other OpenKM servers and refreshed once a day. That means any change to this parameter will take effect tomorrow after restarting the OpenKM service.
rss.news.visible	***Integer***	*Number of visible news items at the same time.* *1*
decimal.pattern	***String***	*Set the decimal pattern.* *.*
sanitize.lucene.fields	***Boolean***	*When enabled, metadata fields are sanitized.* *Because OpenKM metadata groups use characters reserved by Lucene like ":", when they are indexed by Lucene, these fields must be sanitized to be used in Working with Lucene queries.* *After enabling this parameter, you must reindex the whole repository. Take a look at Rebuild indexes.*
search.by.attachment.parent	***Boolean***	*Enable or disable the search for documents in emails (attachments).* *false*
property.group.element.default.access	***Boolean***	*When set to true, all the metadata fields are shown to the users when false, none of the metadata fields are shown.* *Sample scenarios:* • *When the value is set to **true,** all the **metadata fields are shown by default**. In this scenario, it makes sense to use **revokeUser** and **revokeRole**.* • *When the value is set to **false,** all the **metadata fields are not shown by default**. In this scenario, it makes*

		sense to use ***grantUser*** *and* ***grantRole****.* *Additional information at Metadata Security element* *true*

Experimental configuration parameters

Property	*Type*	*Description*
unit.testing.folder	***String***	*The folder used by the testing process.* */okm:root/okmTesting*
unit.testing.user	***String***	*The user is used for unit testing.* *okmAdmin*
unit.testing.password	***String***	*The user's password is used for unit testing.* *admin*
unit.testing.url.base	***String***	*The unit test URL.* *http://localhost:8080/OpenKM/*
rss.news	***Boolean***	*Enable or disable the news on the login page.*
rss.news.box.width	***Integer***	*The width of the news box on the login page.* *300*
rss.news.max.size	***Integer***	*The maximum number of news items retrieved from the news server.* *10* *News is cached from the OpenKM server and refreshed once c change to this parameter will take effect tomorrow after restar service.*
rss.news.visible	***Integer***	*Number of visible news items at the same time.* *1*
check.node.deletion.integrity	***Boolean***	*Enable or disable the node deletion integrity check.*

okm.content.security.policy	***String***	*By default, embedding the application in an iframe is only allowed from th required because it is used in the web UI. This is the default value:*
		`frame-ancestors 'self'`
		The next sample shows how to enable other domains:
		`frame-ancestors 'self' http://example.com`
		*The configuration parameter must be set in the **openkm.prope** OpenKM service must be restarted to take effect.*
		More information about content security policy https://develop US/docs/Web/HTTP/Headers/Content-Security-Policy

okm.cors.configuration.policy	***String***	*By default, CORS is enabled for all origins, methods, and headers. This is*
		`[{"path":"/rest/**", "origins":["*"], "methods":["*"], "headers":["*"]}, {"p ["*"], "methods":["*"], "headers":["*"]}]`
		The next sample shows how to only enable access to the REST API for seve disable others.
		`[{"path":"/rest/**", "origins": ["http://domain.com","https://domain.com","http://domain2.com","h "methods":["*"], "headers":["*"]}, {"path":"/**", "origins":["http://localh ["*"], "headers":["*"]}]`
		*The configuration parameter must be set in the **openkm.prope** OpenKM service must be restarted to take effect.*
		*The String value is a list of objects in JSON format. The order relevant because internally the first match will be applied, wh "/**" should always be applied at the end.*
		More information about CORS at https://developer.mozilla.or US/docs/Web/HTTP/CORS

ldap.ignore.partial.result.exception	***Boolean***	*When AD integration is enabled, warnings of PartialResultException are s true, these warnings are not shown in the log.* `true`
rest.client.call.timeout	***String***	*Set the maximum request timeout in seconds when the OpenKM applicatio*

		to a third-party application. For example, a jbpm-console request.
		15

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.

MIME types

The application supports several document MIME types. Every MIME type has one or more 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 but 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 whose 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.* *If there are multiple extensions, each one should be separated by the character ",".* *epub*
Image	*Image file.*

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

- *Click the **Create button**.*

Edit MIME Type

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

Delete MIME type

- *Click the **Delete icon** .*
- *In the **Delete MIME type view**, click the **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 the 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. Basic server information like memory and disk usage is also shown.

To understand how your repository is growing and to estimate hard disk usage, 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 users' 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 its own trash and every user is responsible for purging their own trash. There's a feature **Administration > Utilities > Purge trashes** that helps with this.*

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.*
- *In the **Pending task queue view**, the status of tasks is shown. 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 it is executed in the background. If the number of documents to process is huge, this task may take hours to complete.

Check the status of the Text extraction queue

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

- *Click the menu option **Pending text extractor queue** at the top right.*

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

- *In **Pending text extractor queue view**, the status of the text extraction process is shown. 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 (for example, if you have added a new text extractor). You can find more information in Database description about how to do it.*

Check the status of the Antivirus check queue

When uploading 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.*
- *In **Antivirus check queue view**, the status of the antivirus analysis process is shown. This is a list of the documents to be processed or in progress.*

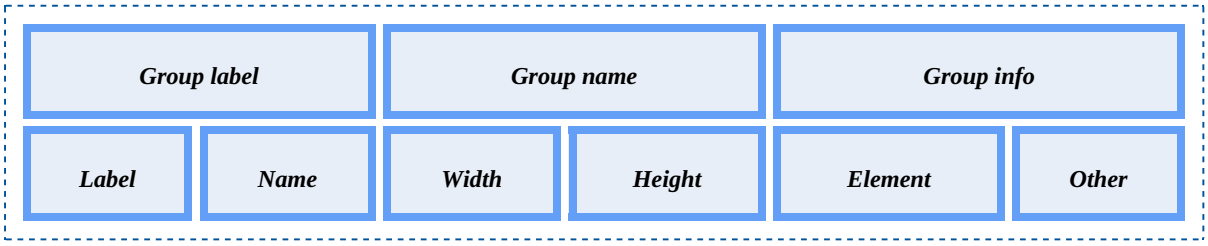
More information is available at Performance configuration parameters.

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	*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.*
Field height	*This parameter is used for rendering the height of the field in the user interface.*
Element	*This parameter sets the element type. It is used to render multiple values in the field in the user interface.* *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 the **{major}** and **{minor}** values are two numbers (major release and minor release). For example, the location for version 3.10 of the DTD is http://www.openkm.com/dtd/property-groups-3.10.dtd*

> *Each application version supports only a couple of DTD definitions. Read Application version compatibility table to see which are supported by your application.*

Basic XML skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>

</property-groups>
```

- *The **DOCTYPE** is a formal definition of a DTD. In the previous example, the DOCTYPE is **property-groups-3.10.dtd**.*
- *The XML begins with **<property-groups>** tag and ends with **</property-groups>**. Within 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**K**M **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.* ***The name must be unique****.* *The name must start with "**okg:**".* *Please use only letters, numbers, and underscore: "0-9a-zA-Z_".*	*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, only from 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*

Basic example

Available field types:

- *Checkbox*
- *Input*
- *Iframe*

- *Select*
- *Separator*
- *Suggestbox*
- *Text*
- *TextArea*

More information about field type elements is available at [Metadata fields]

*It's a 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 the base (changing **okg** to **okp**). That 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, 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 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 into OpenKM

When you register a new metadata definition in OpenKM, the metadata values are stored in two separate tables. One table is for current metadata values and the other is for historical metadata values.

Sample:

```
<?xml version="1.0" encoding="UTF-8"?>
```

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

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <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 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 historical metadata values)*
  - ***RGT_UUID****: ID of the node which has this metadata*
  - ***RGT_VERSION****: ID of the related version*
  - ***RGT_PRO_INPUT1****: Value of property okp:consulting.input1*
  - ***RGT_PRO_INPUT2****: Value of property okp:consulting.input2*
  - ***RGT_PRO_SEPARATOR****: Value of property okp:consulting.separator*

The prefix used for all the columns will be ***RGT_****.*

The column with name ***RGT_UUID*** *contains the value of the node UUID.*

Be careful with the 32-character limitation for Oracle table and column names.

Default column size by type:

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

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

*If you have larger metadata values, you should increase the value of the column using the attribute "**dbColumnSize**". See the sample below:*

*The value of the attribute "**dbColumnSize**" will only be applied the first time you register the metadata group or each time you add a new file. After the column has been created in the database, changing 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 from 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.*
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*
  - *Iframe*
  - *Select*

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

    - *Option*
    - *Validator*
- *Separator*
- *Suggestbox*
    - *Validator*
- *Text*
- *TextArea*
    - *Validator*
- *VerticalPanel*
    - *Checkbox*
    - *Input*
    - *Iframe*
    - *Select*
    - *Separator*
    - *Suggestbox*
    - *Text*
    - *TextArea*
    - *VerticalPanel*
    - *HorizontalPanel*
- *HorizontalPanel*
    - *Checkbox*
    - *Input*
    - *Iframe*
    - *Select*
    - *Separator*
    - *Suggestbox*
    - *Text*
    - *TextArea*
    - *VerticalPanel*
    - *HorizontalPanel*

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

Metadata Checkbox field

Checkbox

Best used to represent boolean data:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata entries cannot 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, the group name "**okg:consulting**" can be used to create the field name "**okp:consulting**.checkbox" 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 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*
description	*General description.*	*false*
readonly	*The set field cannot be modified in the user interface. By default, the value is "**false**".*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.*	

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

	By default, fields are stored in a database column with a size of 128 characters. Because your data may be longer 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 use "**okg:consulting**" as the group; the field name then becomes "**okp:consulting**.iframe" 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.* *There are two parameters set in the request:* • *uuid (the node of the currently selected object in the user interface).* • *propertyGroup (the name of the metadata group).* *If your URL value is http://www.openkm.com, the URL loaded into 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>
```

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 entries 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, when "**okg:consulting**" has been used for the field name "**okp:consulting**.select", changing "**okg**" to "**okp.**" That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*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 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.*	
height	*The height of the HTML element.*	

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

optionsQuery	*A metadata query is used to get the selected options.*	*
className	*A class is used to retrieve the selected options.*	*
suggestion	*The application can automatically suggest values based on document content analysis when adding metadata to documents. By default, the application includes some implementations that can be easily extended.* • ***com.openkm.form.suggestion.DocumentContentContainsSuggestion****: Gets recommendations based on the document's content (uses String.contains)* • ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion****: Gets suggestions based on the document's content (uses StringTokenizer).* • ***com.openkm.form.suggestion.DocumentTermsSuggestion****: Gets recommendations based on the document's term vectors.* *For more information, read Creating your own Suggestion Analyzer.*	*false*
parentElement	*Parent field name value. In this case, the value is hierarchical with other field values (the parent).*	*false*
description	*General description.*	*false*
read-only	*The set field cannot be modified in the user interface. By default, the value is "**false**."*	*false*
dbColumnSize	*Set the size of the database field where the value is stored.* *By default, fields are stored in a database column of 128 characters. Because your data might be longer 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 the 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 (table + optionsQuery).*
- *className.*

One of these options must be defined; otherwise, you will get an error. It cannot be done as a combination of several; only one can be chosen.

Basic select example

- *Type is equal to* ***simple****.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label" name="okp:consulting.select" type="simple">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </property-group>
</property-groups>
```

Allow multiple choices

- *Type is equal to* ***multiple****.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="select label" name="okp:consulting.select" type="multiple">
      <option label="one" value="001" />
      <option label="two" value="002" />
      <option label="three" value="003" />
    </select>
  </property-group>
</property-groups>
```

Based on SQL query

We can also get these option values using 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 retrieve 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 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 are shown in green.

The application includes some implementations that can be easily extended.

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

- ***com.openkm.form.suggestion.DocumentContentContainsSuggestion****: Gets recommendations based on the document's content (uses String.contains)*
- ***com.openkm.form.suggestion.DocumentContentTokenizerSuggestion****: Gets suggestions based on the document's content (uses StringTokenizer).*
- ***com.openkm.form.suggestion.DocumentTermsSuggestion****: Gets recommendations based on the document's term vectors.*

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 option values of the field "**okp:consulting.children**" have a parent relationship with "**okp:consulting.parent**". It's defined by **parentElement="okp:consulting.parent"**.*
- *The option value "**benefits**" of the field "**okp:consulting.children**" has a parent value "**directives**" from the field "**okp:consulting.parent**".*

Tree relationships:

- *directives*
    - *benefits*
    - *eligibility*
    - *services*
- *handbook*
    - *basic*
    - *normal*
    - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
```

```
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="Parent" name="okp:consulting.parent" type="simple">
      <option value="directives" label="Directives" />
      <option value="handbook" label="Handbook Paragraphs" />
    </select>
    <select label="Children" name="okp:consulting.children" type="simple" parentElement="okp:consulting.parent">
      <option value="benefits" label="Benefits" parentValue="directives" />
      <option value="elegibility" label="Eligibility" parentValue="directives" />
      <option value="services" label="Services" parentValue="directives" />
      <option value="basic" label="Basic" parentValue="handbook" />
      <option value="normal" label="Advanced" parentValue="handbook" />
      <option value="advanced" label="Normal" parentValue="handbook" />
    </select>
  </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>
```

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 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, if the metadata group name "**okg:consulting**" has been used, the field name "**okp:consulting.separator**" can be created 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 percentages; for example:* • *width="50%" will work.* • *width="100px" will be ignored in the KCenter UI.*	*false*
readonly	*The field cannot be modified in the user interface. By default, the value is "**false**".*	*false*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

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

Metadata Suggestbox field

This is an implementation of the typical AJAX suggest box component:

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two metadata 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, "**okg:consulting**" has been used for the metadata group and the field name "**okp:consulting.suggestbox**" 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*
height	*The height of the HTML element.*	*false*
description	*General description.*	*false*
filterQuery	*A SQL query to filter the data. Use {0} to place the user input.*	*
valueQuery	*A SQL query to obtain the value of an identifier. Use {0} to place the identifier.*	*
className	*A class is used to retrieve the selected options.*	*

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

filterMinLen	*The minimum number of characters in the input to begin filtering results.*	*true*
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 suggestbox 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 that the user enters to filter the values from the table.

Create COUNTRY table:

```
create table COUNTRY (
  CT_ID varchar(2),
  CT_NAME varchar(32),
  primary key(CT_ID)
);

insert into COUNTRY (CT_ID, CT_NAME) values ('es', 'Spain');
insert into COUNTRY (CT_ID, CT_NAME) values ('pt', 'Portugal');
insert into COUNTRY (CT_ID, CT_NAME) values ('it', 'Italy');
```

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//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 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 property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
  <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
    className="com.openkm.plugin.form.values.SuggestBoxUserList" />
  </property-group>
</property-groups>
```

For more information, read [Creating your own Suggestbox plugin]

Metadata text field

Prints text that can contain HTML tags.

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.* *From version 7.1.41, the label can render HTML. In the case of 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*	*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 the basis for the metadata field name is a good practice. For example, the group name "**okg:consulting**" has been used as the basis for the field name "**okp:consulting**.text" 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*
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*

```
<?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 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 a good practice. For example, "**okg:consulting**" has been used as the group name and the field name as "**okp:consulting**.textarea", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
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 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*
readonly	*The field cannot 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.

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 on Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <textarea label="textarea label" name="okp:consulting.textarea" width="200px" />
  </property-group>
</property-groups>
```

Metadata Validator element

The goal of the Validator element is to ensure that the user provides the necessary and properly formatted information needed to complete an operation successfully.

The validator can only be used in combination with other fields:

- *Metadata Checkbox field*
- *Metadata Input field*
- *Metadata Select field*
- *Metadata Suggestbox field*
- *Metadata TextArea field*

Available validators:

name	Description
req	*Required value.*
alpha	*Alphabetic value.*
alphanum	*Alphanumeric value.*
dec	*Decimal value.*
num	*Numeric value.*
email	*E-mail value.*
url	*URL value.*
maxlen	*Set the maximum length allowed.*
minlen	*Set the minimum length required.*
lt	*Less than.*

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

gt	*Greater than.*
min	*Minimum value.*
max	*Maximum value.*
regexp	*Regular expression.*
plugin	*Create your form validator.* *More information is available 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 ones. 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>
```

Metadata Option element

The option element is only used in combination with [Metadata Select field](). The option value represents one of a list of distinct values 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.* ***The value must be unique****.* *It is 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 the other select.*	*false*
selected	*When present, it specifies that an option should be pre-selected when the form is rendered.*	*false*
cssClass	*CSS class.* *More information about creating your own styles 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">
<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 relationship 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**" from the field "**okp:consulting.parent**".*

Tree relationships:

- *directives*
    - *benefits*
    - *eligibility*
    - *services*
- *handbook*
    - *basic*
    - *normal*
    - *advanced*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <select label="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 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 a good practice. For example, if "**okg:consulting**" has been used for the group name, the field name could be "**okp:consulting**.input", changing "**okg**" to "**okp**". That helps to find the metadata group based on the metadata field name.* *Please use only letters, numbers, and underscores: "0-9a-zA-Z_".*	*true*
type	*Type 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 is 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 %, 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	*The set field cannot be modified in the 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 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 the 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 at Metadata Validator element)*
- ***Security*** *(more information at Metadata Security element)*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input" />
  </property-group>
</property-groups>
```

Metadata VerticalPanel field

A vertical layout that allows you to sort components inside.

Attribute	*Description*	*Required*
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 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 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 the metadata field name. For example, if "****okg:consulting****" has been used for the group name, use the field name "****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 the 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*
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**" which, when set to true, shows all the metadata fields to users; when 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**, all the **metadata fields are not 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 of 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	*Set access to a user.*
grantRole	*Set access to the members of the role.*
revokeUser	*Revoke access to a user.*
revokeRole	*Revoke access to the members of the role.*

Basic grant example

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

- *Grant access to a field for a specific user: only **testUser** 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

The 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**. 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. That means that all users will be able to see the property unless there is a child **revokeUser** tag to exclude users.*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1">
        <revokeUser name="testUser"/>
    </input>
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

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

*Label keys must always start with the "**kcenter.**" string.*

Translations

Register translations in 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
```

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

Metadata definition

Register the metadata definition in Administration > Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
   "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
   <property-group label="kcenter.technology.group" name="okg:technology">
      <input label="kcenter.tecnology.title" type="text" name="okp:technology.title">
         <validator type="req"/>
      </input>
      <input label="kcenter.tecnology.date" type="date" name="okp:technology.date" />
      <textarea label="kcenter.tecnology.description" name="okp:technology.description"/>
      <select label="Type" name="okp:technology.type" type="multiple">
         <option label="kcenter.tecnology.description.newProduct" value="t1" selected="true"/>
         <option label="kcenter.tecnology.description.newFeature" value="t2" selected="true"/>
         <option label="kcenter.tecnology.description.customization" value="t3" />
         <option label="kcenter.tecnology.description.integration" value="t4" />
         <option label="kcenter.tecnology.description.quality" value="t5" />
         <option label="kcenter.tecnology.description.other" value="t6" />
      </select>
      <select label="kcenter.tecnology.priority" name="okp:technology.priority" type="simple">
         <option label="kcenter.tecnology.priority.low" value="low" selected="true"/>
         <option label="kcenter.tecnology.priority.medium" value="Medium"/>
         <option label="kcenter.tecnology.priority.high" value="High" />
      </select>
      <textarea label="kcenter.tecnology.comment" name="okp:technology.comment"/>
      <input label="kcenter.tecnology.link" type="link" name="okp:technology.link"/>
   </property-group>
</property-groups>
```

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

Manage users and roles

In the Manage users and roles view, you can do:

- *Manage users*.
- *Manage roles*.
- *Manage user mail accounts*.
- *Logged-in 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, 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 the top menu* ***Authentication button*** *and then click on the submenu* ***Users link***.

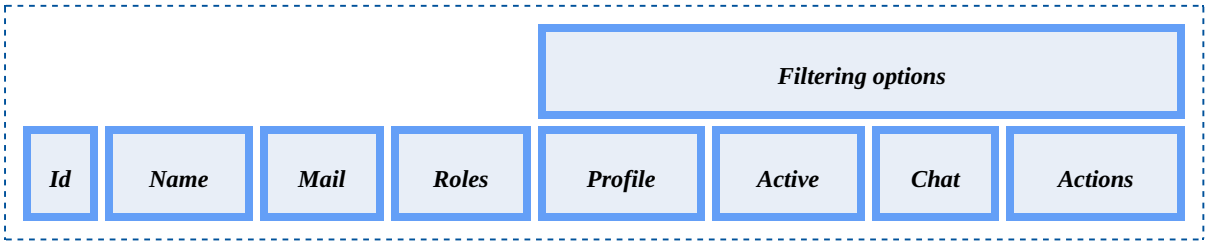
By default, the application manages users and roles in its own database.

It is also possible to connect the application to an external directory (LDAP directory, for instance).

The login module can be extended 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 displayed in a table.



User parameters:

Field	Description	Required
Id	*The id the user uses to log in.* *Two users cannot have the same id.*	true
Name	*User's name.*	true
Mail	*User's email address.*	true
Roles	*Roles - groups the user is a member of.* *A user must belong to at least one of the roles 'ROLE_ADMIN' or 'ROLE_USER'.*	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 whether 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:* • *Edits a user.* • *Deletes a user.* • *Sets the profile.* • *Sets the user's tenant.* • *Configures mail accounts. More information here: Manage user mail accounts.* • *Shows the user's activity log.* • *Deletes the user's 2FA secret.*	*
** Mail account and Twitter account are optional.*		

Create a user

- *At the top right of the table, click on the* ***New user icon****.*
- ***Fill in all the fields****. Remember that all users must belong to at least one of the roles 'ROLE_ADMIN' or 'ROLE_USER'.*
- *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

- *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

- *At 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 to the application repository.

To manage a user mail account**, **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 /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 to be deleted 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 good practice, before clicking the Create button, to* ***check the mail configuration****. There's a button named* ***Check*** *that will verify the mail configuration. If all is correct, the application will get an* ***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 mail in different application folders.

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

For example, filters can help automatically catalog mail based on a "subject" value or "from" value, among others.

Mail account filters are shown in a table structure.

Folder	***Grouping***	***Active***

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 filter enabled or not.*	*true.*
Actions	*Available actions:* • *Edit a filter.* • *Delete a filter.* • *Show rules.*	

At least one 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.

Field	***Operation***	***Value***	***Active***

Mail account rule parameters:

Field	*Description*	*Required*
Field	*Sets the mail field that will be evaluated by the rule.* *Available options are:* • *From* • *To* • *Subject* • *Content*	*true.*
Operation	*Set the operation that will be used 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 rule enabled or not.*	*true.*

Create a mail account filter

- *Click on the* ***Show filters icon***.
- *On top right click on* ***New filter icon***.
- ***Fill in all the fields***.
- *Click on the* ***Create button***.

Edit a mail account filter

- *Click on the* ***Show filters icon***.
- *On top right click on* ***Edit filter icon***.
- ***Modify the fields***.
- *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 the top menu* ***Authentication button*** *and then click the* ***Roles link*** *in the submenu.*

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.

Id	***Active***	***Actions***

Role parameters:

Field	***Description***	***Required***
Id	*The id of the role.* *Two roles cannot have the same id.*	*true*
Active	*Sets whether a role is enabled or not.*	*true*
Action	*Available actions:* • *Edit a role.* • *Delete a role.* • *Set a role tenant.*	

Create a role

- *On the top right in the table, click on the* ***New role icon****.*

- ***Fill all the fields***.
- *Click on the* ***Create button***.

Edit a role

- *Click on the* ***Edit role icon***.
- ***Modify the fields***.
- *Click on the* ***Edit button***.

> *ROLE id cannot be modified to prevent repository security inconsistencies. This operation can only be done at the database level.*

Delete a role

- *Click on the* ***Delete role icon***.
- *Click on the* ***Delete button***.

Export the roles

- *On the top right, click 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 structure:

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

The 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.

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.	*

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

To Users	*When present, indicates a set of users to send the message to.*	*
To Roles	*When present, indicates a set of users who are members of some 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

- *At the top right, click on* ***Clear button***.

This operation cannot be reverted. All messages in the queue will be deleted.

Refresh the message queue

- *At the top right, click on* ***Refresh button***.

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 to show only the search view.*

The application, and especially the user interface, come by default with several features and extensions.

Not all users need access to everything, especially at the beginning when many features can cause confusion for end users.

Our suggestion is to go from fewer to more features as users gain knowledge of 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*
Web skin	*Sets the default skin (CSS styles applied).* *Info message*
Keywords enabled	*Enables or disables keyword shortcuts.*
Increase version	*When enabled, users can choose whether to upload a document as a minor or 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 event notifications	*When enabled, messages sent by the application are shown to the user in a pop-up.*

	Common notifications: • *New documents uploaded.* • *New task assigned.* • *New mails received.*
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.*
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.*

Pagination section

Pagination control

Field	*Description*
Page list	*Sets the distinct values that will be shown for the number of elements per page.* *For example, the value "10;25;50;100" sets the 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.*

	To separate the distinct values, use the character ";".
Type filter	*When enabled, a type filter is shown, which helps the user display only documents, folders, records, or mails. *****
Show folders by default	*Shows or hides folders by default. *****
Show documents by default	*Shows or hides documents by default. *****
Show mails by default	*Shows or hides mails by default. *****
Show records by default	*Shows or hides records by default. *****

** When the type filter is enabled, options such as 'Show folders by default' must be active; otherwise they are not taken into consideration and all types are shown.*

Section General

Shortcuts are keyword combinations for a faster way of accomplishing a task.

Field	*Description*
Rename	*Executes the rename action on the selected node.* *F2*
Copy	*Executes 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*
Testing	*Shows a pop-up to do some user interface 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	*Logs the user out of the application.* *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.*
Templates	*Shows or hides the templates in the stack panel.*
Personal	*Shows or hides personal information 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 the trash bin in the stack panel.*

Dashboard section

Field	*Description*
User	*Shows or hides the user 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.*

Tab section

Field	*Description*
	Sets the default tab visible when the user logs in to the application.

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

Default tab	*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.* *The 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 downloading the document version.*

Preview	*Shows or hides the preview tab.*
Metadata	*Shows or hides the metadata tabs.*
Relation	*Shows or hides the relations tab.*

Tab folder section

The tab folder is visible when a folder is selected in desktop view.

Field	*Description*
Properties	*Shows or hides the folder properties tab.*
Security	*Shows or hides the folder tab.*
Notes	*Shows or hides the notes tab.*
Preview	*Shows or hides the preview tab.*
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.*

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

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 in to the application.*
Security	*Shows or hides the record 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 search section

Field	*Description*
Default search	*Sets the default tab visible in the search view.* *Available options:* • *Simple* • *Advanced*
Simple	*Shows or hides the simple search view.*

Advanced	*Shows or hides the advanced search view.* *The advanced search view is composed of:* • *Tab basic* • *Tab advanced* • *Tab metadata*
Tab basic	*Shows or hides the basic tab in the 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.*

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

The tab advanced contains the advanced fields for searching.

Field	*Description*
Folder	*Shows or hides the folder selector.*
Category	*Shows or hides the category selector.*
Type	*Shows or hides the node type.* *In 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 or contextual menu options.

Hierarchy:

- *Menu*
    - *Toolbar*

> *Preference hierarchy: higher-level preferences take precedence over lower-level ones. If an upper-level option is disabled, it will also be disabled at a lower level.*
>
> *For example, when the download menu option is disabled, it will not be shown in 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.*
Bookmarks	*Shows or hides the bookmark menu.*
Templates	*Shows or hides the templates menu.*

Menu bookmark section

Field	*Description*
Add bookmark	*Shows or hides the bookmark menu option.*
Go home	*Shows or hides the go-to home menu option.* *When 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*
Preferences	*Shows or hides the user preferences menu option.*

Menu help section

Field	*Description*
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.*
Find records	*Shows or hides the find records menu option.*
Go folder	*Shows or hides the go to folder menu option.* *This option, for example, 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.*

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

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 option to purge all user trash.*
Restore	*Shows or hides the restore from trash menu option.*
Refresh	*Shows or hides the option to refresh the user interface.*
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.*
Send document attachment	*Shows or hides the send document attachment 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 option to update a document.*
Edit	*Shows or hides the option to edit a document.*

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

Cancel Edit	*Shows or hides the option to cancel editing a document.*
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.*
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.*

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

Profile Notification tab

The Application sends notifications based on events that occur on users' subscriptions. These notifications can be sent as an immediate notification email or as all the messages grouped into 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

*By default, **these two groups are empty, and no notifications will be sent**. If you want to use notifications, you must 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.*
MOVE_DOCUMENT	*Move a document.*
PURGE_DOCUMENT	*The document is 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_DESCRIPTION	*Set the description*

Folder

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

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 removed from the repository.*
RENAME_FOLDER	*Change the name of a folder.*
SET_FOLDER_DESCRIPTION	*Set the description*

Mail

Notification	*Description*
CREATE_MAIL	*Create a new email.*
GET_MAIL_PROPERTIES	*Get the mail properties.*
DELETE_MAIL	*Delete a mail.*
PURGE_MAIL	*The mail is removed from the repository.*

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

RENAME_MAIL	*Change the name of a mail.*
MOVE_MAIL	*Move a mail.*
COPY_MAIL	*Copy a mail.*
GET_CHILDREN_MAILS	*Get the mails.*
SET_MAIL_NODE_CLASS	*Set the mail NodeClass*
SET_DESCRIPTION	*Set the description*
SET_MAIL_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.*
FORCE_UNLOCK_RECORD	*Force unlock a record.*
LOCK_RECORD	*Lock a record.*
PURGE_RECORD	*The record is removed from the repository.*
RENAME_RECORD	*Change the name of a record.*

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

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_DESCRIPTION	*Set the description*

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 mail	*Shows or hides the find mail menu option.*
Find records	*Shows or hides the find records menu option.*
Download	*Shows or hides the download document menu option.*
Download PDF	*Shows or hides the download document as a PDF 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.*
Add record	*Shows or hides the add record menu option.*
Edit	*Shows or hides the menu option to edit a document.*

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

Update	*Shows or hides the menu option to update a document.*
Cancel Edit	*Shows or hides the menu option to cancel editing a document.*
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.*
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.*
History	*Shows or hides the history menu option.*

File browser section

Sets the file browser columns that are shown.

Field	*Description*
Status	*Shows or hides the status column.*
Icon	*Shows or hides the icon column.*
Name	*Shows or hides the name column.*
Title	*Shows or hides the title column.*

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

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.*

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, emails, records). Relation type definitions are set in relation types.

There are three types of relations:

- *Bidirectional*
- *Parent-child*
- *Many-to-many*

Relation types are shown in a table structure.

Type	***Title***	***Title parent-to-child***	***Title child-to-parent***

Types

Bidirectional

This relation is always set between two nodes.

A is linked to B just 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, just as B is a translation of A.

NODE A --------> <-------- ***NODE B***

Parent - child

This relelation is always set between two nodes.

A is linked to B and B is linked to A, but the relation that links both elements is different.

An example of this relation type is the "Invoice - Quote" relation. B is an invoice of A. A is a quote of B.

QUOTE A --------> ***INVOICE 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

- *At the top right, click on* ***New relation type icon****.*
- ***Choose one type*** *(bidirectional, parent-child or many-to_many).*
- ***Fill in the fields****.*
- *Click on the* ***Create button****.*

For the bidirectional type, you need to fill in the "Title" field.

For the parent-child type, you need to fill in the fields "Title parent-to-child" and "Title child-to-parent".

For the many-to-many type, you need to fill in the "Title" field.

Edit a relation

- *Click on* ***Edit a relation icon****.*
- ***Modify the fields****.*
- *Click on 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 on 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 Profile General tab.

This helps the administrator change the look and feel of the application.

For example, change the font size for Cyrillic or logographic languages.

CSS is shown in a table structure.

Name	***Context***	***Active***	***Actions***

Create a new CSS

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.*
  - *Context: available options are 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 in CSS should take into account which UI will be used by the users.

Classical UI samples (frontend)

Large font sample

```
body, td, a, p, div, input, input.submit , textarea, select, button{
 font-family: Tahoma, Arial, Helvetica, sans-serif;
 font-size: 13px;
}

.gwt-MenuItem, .okm-TopMenuBar .gwt-MenuItem-selected, .okm-Input, .okm-TextArea, .okm-Select {
 font-family: Tahoma, Arial, Helvetica, sans-serif;
 font-size: 13px;
}

.okm-Popup .Caption, .okm-Popup-Error .Caption, .okm-Image-Popup .Caption {
 font-family: Tahoma, Arial, Helvetica, sans-serif;
 font-size: 15px;
}
```

Medium font sample

```
body, td, a, p, div, input, input.submit , textarea, select, button{
 font-family: Arial, Helvetica, sans-serif;
 font-size: 12px;
}

.gwt-MenuItem, .okm-TopMenuBar .gwt-MenuItem-selected, .okm-Input, .okm-TextArea, .okm-Select {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 12px;
}

.okm-Popup .Caption, .okm-Popup-Error .Caption, .okm-Image-Popup .Caption {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 14px;
}
```

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

Default font sample

```
body, td, a, p, div, input, input.submit , textarea, select, button{
 font-family: Arial, Helvetica, sans-serif;
 font-size: 11px;
}

.gwt-MenuItem, .okm-TopMenuBar .gwt-MenuItem-selected, .okm-Input, .okm-TextArea, .okm-Select {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 11px;
}

.okm-Popup .Caption, .okm-Popup-Error .Caption, .okm-Image-Popup .Caption {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 13px;
}
```

New UI samples (KCenter)

Large font sample

```
body, td, a, p, div, input, textarea, select, b-tab {
  font-size: 1.2rem !important; // also can be used 14px;
  font-weight: 400; // Standard weight
}
```

Medium font sample

```
body, td, a, p, div, input, textarea, select, b-tab {
  font-size: 1.1rem !important; // also can be used 13px;
  font-weight: 400; // Standard weight
}
```

Arabic sample

The following example will work from OpenKM version 7.1.31 and later.

For Arabic, several fonts can be used:

- *"Noto Sans Arabic"*
- *"Noto Naskh Arabic"*
- *"Noto Kufi Arabic"*
- *"IBM Plex Sans Arabic"*
- *"Tajawal"*

```
/*
 * Notes about some attribute values:
 * font-size: 1.2rem !important;      // Also can be used 14px;
 * font-weight: 400;                  // Standard weight value
```

```
 */
body, td, a, p, div, input, textarea, select, b-tab, button, ion-app, ion-menu, ion-header, ion-item, ion-content, ion-foo
  font-size: 1.2rem !important;
  font-weight: 400;
  font-family: "Noto Kufi Arabic" !important;
}

/* Properties */
li, td, th {
  font-size: 1.2rem !important;
}

/* breadcrumb */
li.breadcrumb-item.mt-1 a {
  font-size: 1.2rem !important;
}
#rootNodes div {
 min-height: 35px !important;
}

/* navbar */
#okm-mainMenu .navbar-nav > li.active > a {
 padding-top: 12px !important;
 padding-bottom: 11px !important;
    line-height: unset !important;
}

#okm-mainMenu .navbar-nav > li > a {
 padding-top: 12px !important;
    line-height: unset !important;
}

/* search */
.okm-forceEmptyLabel-topPadding {
 padding-top: 18px !important;
}

/* button */
.btn-group.okm-small-button > button {
 height: 30px !important;
}
```

Reports

The application includes the JasperReports engine. This helps administrators create their own reports.

The JasperReports Library is one of the world's most popular reporting engines. It is able to use data coming from any type of data source and produce pixel-perfect documents that can be viewed, printed or exported in a variety of document formats including HTML, PDF, Excel, OpenOffice and Word.

More information about how to build your own reports is available at Creating your own reports.

More information about the JasperReports 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, on Ubuntu, the package to install is **ttf-mscorefonts-installer**.*

```
$ apt-get install ttf-mscorefonts-installer
```

*On 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 script 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* ***Execute report icon.***
- ***Fill in the fields*** *(when the report has 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.

Reports pack

Most of these reports are based on the Activity log events audit trail.

For more information Activity log.

Template reports

Name	Use Activity log	Required parameters	Description
JavaReportPattern.jrxml	No.	No.	A template pattern that uses Java to retrieve data.
SQLReportPattern.jrxml	No.	No.	A template pattern that uses SQL to retrieve data.

Authentication

Name	Use Activity log	Required parameters	Description
-	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	Use Activity log	Required parameters	Description
DocumentCanceledCheckout.rep	Yes.	Yes.	Lists documents that canceled the edition, filtered by date range.

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

DocumentCheckin.rep	*Yes.*	*Yes.*	*Lists documents updated, filtered by date range.*
DocumentCheckout.rep	*Yes.*	*Yes.*	*Lists documents being edited, filtered by date range.*
DocumentCreate.rep	*Yes.*	*Yes.*	*Lists documents created, filtered by date range.*
DocumentDelete.rep	*Yes.*	*Yes.*	*Lists documents deleted, filtered by date range.*
DocumentGetChildren.rep	*Yes.*	*Yes.*	*Shows when users list the 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 content of the document.*
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 the user retrieved the properties of a document. Results are filtered by date range.* *For example, when the document is selected, it internally calls a method named "getProperties" 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.*	*Lists the fields of a metadata group.* *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 updated, filtered by date range.*
DocumentMove.rep	*Yes.*	*Yes.*	*Lists documents moved, filtered by date range.*
DocumentPurge.rep	*Yes.*	*Yes.*	*Lists documents purged, filtered by date range.* *When a document is purged, it will disappear from the application repository.*
DocumentRenamed.rep	*Yes.*	*Yes.*	*Lists documents renamed, filtered by date range.*
DocumentUnlock.rep	*Yes.*	*Yes.*	*Lists documents unlocked, filtered by date range.*
DocumentsAccessibleByUserOrdered.jrxml	*No.*	*No.*	*Retrieve a list of available documents - security filtered - ordered by folder name.* *Optimized for CSV format.* *Not recommended for large*

			repositories; it can take a lot of time.
DuplicatedFiles.jrxml	*No.*	*No.*	*List of duplicated files in the application repository, based on document checksums.*
ListDocumentsWithChecksum.rep	*No.*	*Yes.*	*List of all documents in some folders and subfolders with checksum values.*

Folders

Name	*Use Activity log*	*Required 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.*
FolderGetChilden.rep	*Yes.*	*Yes.*	*Shows when users list the 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 the folder depth, for example before starting the folder ZIP download.* *There's an internal method named "getContentInfo" that does it.*

FolderGetProperties.rep	*Yes.*	*Yes.*	*Shows when the user retrieved the properties of a folder. Results are filtered by date range.* *For example, when the folder is selected, it internally calls a method named "getProperties" to fill the contents of the properties tab in the user interface.*
FolderMove.rep	*Yes.*	*Yes.*	*Lists folders moved, filtered by date range.*
FolderPurge.rep	*Yes.*	*Yes.*	*Lists folders purged, filtered by date range.* *When a folder is purged, it will disappear from the application repository.*
FolderRename.rep	*Yes.*	*Yes.*	*Lists folders renamed, filtered by date range.*
FolderSecurity.rep	*No.*	*Yes.*	*Lists folders and subfolders showing security for roles and/or users.*
FolderGetSize.rep	*No.*	*Yes.*	*Lists folders and subfolders, showing the size of each.*

Mails

Name	*Use Activity log*	*Required parameters*	*Description*
MailCreate.rep	*Yes.*	*Yes.*	*Lists mails 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 purged, filtered by date range.*

			When a mail is purged, it will disappear from the application repository.
MailRename.rep	*Yes.*	*Yes.*	*Lists mails renamed, filtered by date range.*
MailMove.rep	*Yes.*	*Yes.*	*Lists mails moved, filtered by date range.*
MailCopy.rep	*Yes.*	*Yes.*	*Lists mails copied, filtered by date range.*
MailGetChilden.rep	*Yes.*	*Yes.*	*Shows when users list the 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

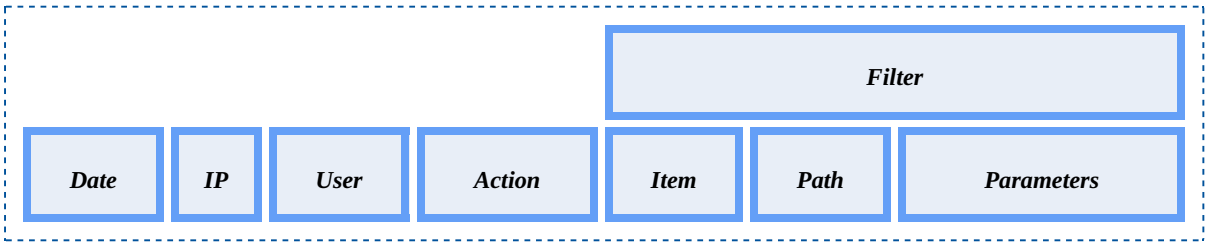
Name	*Use Activity log*	*Required parameters*	*Description*
TenantUserQuota.jrxml	*No.*	*No.*	*Document size (in MB) by user and tenant.*
RegisteredUsers.jrxml	*No.*	*No.*	*Registered users.*

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

Activity log

This application provides 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 date.*
GRANT_USER	*Adds a user grant on a node.*
REVOKE_USER	*Removes the 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 the 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.*

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.*
UNLOCK_DOCUMENT	*Unlocks a document.*
DOWNLOAD_DOCUMENT	*Downloads a document.*
DOWNLOAD_PREVIEW	*Previews a document.*

Folder actions

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 in it).*
GET_FOLDER_PROPERTIES	*Retrieves all folder properties.* *It 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 removed from the application.*
RENAME_FOLDER	*Renames a folder.*
EXPORT_FOLDER_AS_JAR	*Folder contents are exported as a JAR file.*

Mail actions

Field / Property	*Description*
CREATE_MAIL	*Creates a mail.*
GET_MAIL_PROPERTIES	*Retrieves all mail properties.* *It 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 removed from the application.*
RENAME_MAIL	*Renames a mail.*
MOVE_MAIL	*Moves a mail.*
COPY_MAIL	*Copies a mail.*
GET_CHILDREN_MAILS	*Retrieves the mail list in a node.* *List of mail in a folder or record.*
DOWNLOAD_MAIL	*Downloads a mail.*
SEND_MAIL_LINK	*Sends an email with a node link.*
SEND_MAIL_ATTACHMENT	*Sends an email with document attachments.*
CREATE_MAIL_ATTACHMENT	*Creates an email 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 an email.*

Record actions

Field / Property	*Description*
CREATE_RECORD	*Creates a record.*
GET_RECORD_PROPERTIES	*Retrieves all record properties.* *It 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.*
MOVE_RECORD	*Moves a record.*
COPY_RECORD	*Copies a record.*
GET_CHILDREN_RECORDS	*Retrieves a list of records in a node.* *A list of records in a folder or record.*
SET_RECORD_TITLE	*Sets the title.*

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 a single metadata group property of a node (document, folder, mail, or record).*

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 ZIP.*
RESTORE_NODE_VERSION	*Restores the version of a node (document, folder, or record).*

Repository actions

Field / Property	*Description*
PURGE_TRASH	*Trash is purged.* *All related nodes are removed from the application.*

Admin actions

These actions are only done by users with Administration grants.

Field / Property	*Description*

ADMIN_ACTIVITY_LOG	*The administrator uses "activity log".*
ADMIN_ACTIVE_SESSIONS	*The administrator uses Logged users.*
ADMIN_USER_CREATE	*Creates a user.*
ADMIN_USER_EDIT	*Edits a user.*
ADMIN_USER_DELETE	*Deletes a user.*
ADMIN_USER_ACTIVE	*Enables or disables a user.*
ADMIN_ROLE_CREATE	*Creates a role.*
ADMIN_ROLE_EDIT	*Edits a role.*
ADMIN_ROLE_DELETE	*Deletes a role.*
ADMIN_ROLE_ACTIVE	*Enables or disables a role.*
ADMIN_CHECK_EMAIL	*Checks the email poll.* *The application reads email accounts to import emails.*
ADMIN_CONFIG_CREATE	*Creates a configuration parameter.*
ADMIN_CONFIG_EDIT	*Edits a configuration parameter.*
ADMIN_CONFIG_DELETE	*Deletes a configuration parameter.*
ADMIN_CRONTAB_CREATE	*Creates a crontab task.*

ADMIN_CRONTAB_EDIT	*Edits a crontab task.*
ADMIN_CRONTAB_DELETE	*Deletes a crontab task.*
ADMIN_CRONTAB_EXECUTE	*Executes a crontab task.*
ADMIN_DATABASE_QUERY	*Executes a database query.*
ADMIN_DATABASE_UPDATE	*Executes a database update query.*
ADMIN_LANGUAGE_CREATE	*Creates a new language.*
ADMIN_LANGUAGE_EDIT	*Edits a language.*
ADMIN_LANGUAGE_DELETE	*Deletes a language.*
ADMIN_LANGUAGE_IMPORT	*Imports a language.*
ADMIN_LOGCAT_LIST	*The administrator uses the logcat tool.*
ADMIN_LOGCAT_VIEW	*The administrator uses the logcat tool.*
ADMIN_LOGGED_USERS	*The administrator uses Logged users.*
ADMIN_MAIL_ACCOUNT_CREATE	*Creates a mail account.*
ADMIN_MAIL_ACCOUNT_EDIT	*Edits a mail account.*
ADMIN_MAIL_ACCOUNT_DELETE	*Deletes a mail account.*
ADMIN_MAIL_ACCOUNT_CHECK	*Checks a mail account.*

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

ADMIN_MAIL_FILTER_CREATE	*Creates a mail account filter.*
ADMIN_MAIL_FILTER_EDIT	*Edits a mail account filter.*
ADMIN_MAIL_FILTER_DELETE	*Deletes a mail account filter.*
ADMIN_MAIL_FILTER_RULE_CREATE	*Creates a mail account rule.*
ADMIN_MAIL_FILTER_RULE_EDIT	*Edits a mail account rule.*
ADMIN_MAIL_FILTER_RULE_DELETE	*Deletes a mail account rule.*
ADMIN_MIME_TYPE_CREATE	*Creates a new MIME type.*
ADMIN_MIME_TYPE_EDIT	*Edits a MIME type.*
ADMIN_MIME_TYPE_DELETE	*Deletes a MIME type.*
ADMIN_USER_PROFILE_CREATE	*Creates a profile.*
ADMIN_USER_PROFILE_EDIT	*Edits a profile.*
ADMIN_USER_PROFILE_DELETE	*Deletes a profile.*
ADMIN_PROPERTY_GROUP_REGISTER	*Registers a metadata group.*
ADMIN_PROPERTY_GROUP_LIST	*Lists metadata groups.*
ADMIN_REPORT_CREATE	*Creates a report.*
ADMIN_REPORT_EDIT	*Edits a report.*

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

ADMIN_REPORT_DELETE	*Deletes a report.*
ADMIN_REPORT_EXECUTE	*Executes a report.*
ADMIN_REPOSITORY_SEARCH	*Performs a Lucene search.*
ADMIN_REPOSITORY_REINDEX	*Reindexes the Lucene repository.*
ADMIN_REPOSITORY_UNLOCK	*Unlocks a node from a repository view.*
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 the repository view.*
ADMIN_REPOSITORY_REMOVE_CONTENT	*Purges content.*
ADMIN_REPOSITORY_REMOVE_CURRENT	*Deletes the current.*
ADMIN_USER_CONFIG_EDIT	*Changes a user profile.*
ADMIN_SCRIPTING	*Executes scripting.*
ADMIN_MESSAGE_QUEUE_CREATE	*Creates a new message.* *For more information, see Message queue.*
ADMIN_MESSAGE_QUEUE_UPDATE	*Updates a message.*
ADMIN_MESSAGE_QUEUE_DELETE	*Deletes a message.*

ADMIN_DELETE_MESSAGE	*Deletes a message.*

Misc actions

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.* *The order of rules*
Name	*Rule description.*
Events	*Available events:* • *Document creation.* • *Document update.* • *Document delete.* • *Document rename.* • *Document purge.* • *Document move.* • *Document version restore.* • *Document download from UI.* • *Document download from UI for preview.* • *Document virus detected.* • *Folder create.* • *Folder delete.* • *Folder move.* • *Export folder as JAR.* • *Mail creation.* • *Mail delete.* • *Mail move.* • *Mail download from UI.*

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

- *Mail import attachments.*
- *Record creation.*
- *Record delete.*
- *Record move.*
- *Record set node class.*
- *Add a metadata group.*
- *Set metadata group.*
- *Remove the metadata group.*
- *Text extraction.*
- *Export several nodes as a ZIP.*
- *User creation.*
- *User login.*
- *User logs out.*
- *Note creation.*
- *Note update.*
- *Note delete.*

At

The evaluation time.

Available values:

- *Pre.*
- *Post.*

Understanding the stages of 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.

- *At the "pre" stage, the rules are triggered when the document has not yet been created.*
- *At the "post" stage, the rules are triggered after the document has been created.*

Exclusive	*When enabled, it indicates that 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 good practice to set first the validations that are least likely to fail.* *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 from the Plugins view. The application's plugin architecture helps administrators add new features to an existing application.

By default, the application comes with built-in plugins:

- *Built-in Automation Validator plugins.*
- *Built-in Automation Action plugins.*

Also, you can create your own plugins:

- *Creating your own Automation Validation.*
- *Creating your own Automation Action.*

More information about how to register a new plugin.

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

Creating a rule

Rules are composed of:

- *General fields.*
- *Actions and validator definitions.*

Creating a rule has two steps. First, define the general fields, then the actions and validation definitions.

- *At 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 as needed.*
  - *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 the* ***Edit button****.*

Edit action and validation definitions

- *Click on* ***Definition icon****.*
- *Modify actions and validations as needed.*
  - *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****.*

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

- *Click on* ***Delete action icon***.

Deleting a rule

- *Click on* ***Delete rule***.
- *Click on the* ***Delete button***.

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

Built-in Automation Validator plugins

Name	*Apply at*	*Parameters*	*Description*
ContainsFolder	*Post*	*Folder name*	*Valid when the node path contains a folder name.* *The value of the parameter folder name is **test**.* *Node with path **/okm:root/folder1** triggered the event.* *The folder **/okm:root/folder1/test** exists, then the validation result is **valid**.* *The value of the parameter folder name is **test**.* *Node with path **/okm:root/folder1** triggered the event.* *A folder **/okm:root/folder1/test** doesn't exist, so th 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*		

			Valid when the node has been created from a template.
IsNotCreatedFromTemplate	*Pre,* *Post*		*Valid when the node has NOT been created from a template.*
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** triggere the event.* *Folder **sometest** contains the value **test**, then the validation result is **valid**.* *The value of the parameter name is **test**.* *Node with path **/okm:root/folder/somefolder** triggered the event.* *Folder **somefolder** does not contain the value **test**, then the validation result is **not valid**.*
NameMatches	*Pre,* *Post*	*Regex*	*Valid when the node name matches the value.*
PathContains	*Post*	*Path*	*Valid when the node path starts with a path.* *The value of the parameter path is **/okm:root/folder1**.*

			*Node with path **/okm:root/folder1/test/temporary** triggered th event.* ***/okm:root/folder1/test/temporary** starts with path **/okm:root/folder1**, then the validation result i **valid**.* *The value of the parameter path is **/okm:root/folder1**.* *Node with path **/okm:root/test/temporary** triggered the event.* ***/okm:root/test/temporary** does not start with path **/okm:root/folder1**, then the validation result i **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**, then 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**, then the validation return is **valid**.* *The value of the parameter folder name is **folder1**.* *Node with path **/okm:root/test/temporary** triggered the event.* ***/okm:root/test/temporary** does not contain a folder named **folder1**, then 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 th event.* *The parent path is **/okm:root/parent-folder**; therefore, the validation result is **valid**.* *The value of the parameter folder name is **folder1**.* *Node with path: **/okm:root/folder2/parent-folder/folder1** triggered the event.* *The parent path is **/okm:root/folder2/parent-folder** Within this path, there is a folder called "folder1"; therefore, the validation result is **valid**.* *The value of the parameter folder name is **folder1**.* *Node with path: **/okm:root/parent-folder/another-folder/folder1/test** triggered the event.* *The parent path is **/okm:root/parent-folder/another-folder**. Within this path, there does not exist a folder named "**folder1**"; therefore, the validation result is **invalid**.* *The value of the parameter folder name is **folder1**.* *Node with path: **/okm:root/parent-folder/temporary** triggered the event.* *The parent path is **/okm:root/parent-folder**. Within this path, there does not exist a folder named "**folder1**"; therefore, the validation result is **invalid**
UserHasRole	*Pre,* *Post*	*Role name*	*Valid when the user has the role assigned.*

Built-in Automation Action plugins

Name	*Apply at*	*Parameter*	*Description*	*Res*
AddCategory	*Post*	*Category*	*Adds a category to the node that triggered the event.*	*App to a typ*
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.*	*App to doc*
AddKeyword	*Post*	*Keyword name*	*Adds a keyword to the node that triggered an event.*	*App to a typ*
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.*	*App to doc*
AddPropertyGroup	*Post*	*Metadata group name*	*Adds the metadata group.*	*App to a typ*
AddPropertyGroupToWizard	*Post*	*Metadata group name*	*Shows a metadata group 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.*	*App to doc*

ExecuteScripting	*Pre* *Post*	*Code*	*Executes a script.*	*App to a typ*
GrantRole	*Post*	*Role name* *Permission*	*Grants a role permission to the node that triggered the event.*	*App to a typ*
GrantUser	*Post*	*User name* *Permission*	*Grants a user permission to the node that triggered the event.*	*App to a typ*
MoveToAZFolder	*Post*	*Folder destination* *Folder depth*	*Creates a folder hierarchy based on the name of the node 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*	*App to a typ*
MoveToFolder	*Post*	*Folder destination*	*Move the node to a new location.*	*App to a typ*
MoveToYearMonthDayFolder	*Post*	*Folder destination*	*Move the node to a new location with a subfolder structure year/month/day.* *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.pdf*	*App to a typ*

RevokeAllRoles	*Post*		*Revokes all roles granted to the node that triggered the event.*	*App to a typ*
RevokeAllUsers	*Post*		*Revokes all users granted to the node that triggered the event.*	*App to a typ*
RemoveCategory	*Post*	*Category*	*Removes the category from the node that triggered the event.*	*App to a typ*
RemoveKeyword	*Post*	*Keyword name*	*Removes the keyword from the node that triggered the event.*	*App to a typ*
RevokeRole	*Post*	*Role name*	*Revokes the role granted to the node that triggered the event.*	*App to a typ*
RevokeUser	*Post*	*User name*	*Revokes the user granted to the node that triggered the event.*	*App to a typ*
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 notification messages. For more information, take a look at Notification configuration parameters.*	*App to a typ*
SetAutoStartToWizard	*Pre, Post*		*Sets the autostart for the wizard.*	*App to a typ*

SetInQuarantine	*Post*		*When the antivirus engine detects a virus in a document, it moves the document into the quarantine folder /okm:root/quarantine.*	*Ap to doc*
SetLanguage	*Post*	*Language*	*Sets the language of the document node that triggered the event.* *Language value is based on ISO 639-1.* *For example, for Spanish the value is "es", for English "en", etc.*	*Ap to doc*
SetSummarization	*Post*		*Sets the document summarization.* *The KEA Summarization service must be enabled.*	*Ap to doc*
SetTextExtracted	*Post*	*String*	*Replaces the original extracted text content of the document node that triggered the event.*	*Ap to doc*

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.

Typically, crontab jobs are automated application maintenance or administration tasks.

The Crontab is shown in a table structure.

Name	Expression	Mime	File name	Mail	Last begin	Last end	Active

Crontab parameters

Field	Description	Required
Name	Job name.	true
Expression	Sets when the job is executed. 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 you consider the backup process to be 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 specify several email addresses using a comma "," as a separator.	false
Last begin	Shows the last time a process started.	-
Last end	Shows the last time a process was completed.	-
Active	Enables or disables process execution.	true

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

Understanding expression syntax

More information: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/scheduling/support/CronSequenceGenerator.htm

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of month or day of week) 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”€ day of month (1 - 31)
 â”‚ â”‚ â””â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€ hour (0 - 23)
 â”‚ â””â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€â”€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”€ 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 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.*

*"0 0 9-17 * * MON-FRI" = on the hour, nine-to-five on weekdays.*

"0 0 0 25 12 ?" = every Christmas Day at midnight.

Create a new crontab job

> *Crontab jobs are based on a plugin architecture.*
>
> *More information at Creating your own Crontab plugin.*

Edit a crontab job

- *Click* ***Edit crontab job icon***.
- ***Modify the fields***.
- *Click the* ***Edit button***.

Execute a crontab job

- *Click* ***Execute crontab job icon***.

Delete a crontab job

> *Crontab jobs are 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*
Dashboard Activity Purge	*0 0 0 * * **	*Executes daily at 00:00.*	*Cleans dashboard activity logs older than one month.*
User Mail Importer	*0 */30 * * * *	*Executes every 30 minutes.*	*Imports emails from registered users' mail accounts.*
Text Extractor Worker	*0 */5 * * * *	*Executes every 5 minutes.*	*Processes the text extraction queue.* *For more information, see Statistics.*
Antivirus Checker Worker	*0 */5 * * * *	*Executes every 5 minutes.*	*Processes the antivirus check queue.* *For more information, see Statistics.*
Process Pending Tasks	*0 */5 * * * *	*Executes every 5 minutes.*	*Executes pending tasks.* *For more information, see Statistics.*
Purge Download Preview Tokens	*0 */5 * * * *	*Executes every 5 minutes.*	*Deletes expired preview tokens that have not been consumed.*
Repository Info	*0 0 0 * * **	*Executes daily at 00:00.*	*Updates repository stats.*
Text Extractor Issues	*0 0 0 * * **	*Executes daily at 00:00.*	*Collects text extraction issues and sends an email warning to the administrator.*
SyncLdapUsers	*0 0 0 1 1*	*Executes once a*	*Synchronizes users and roles from the AD to the OpenKM*

	*	*week.*	*database.*

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

Languages

The application can work with several languages. The language terms are defined here.

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 completed 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* ***new language icon****.*
- *Fill the fields*
  - *The id must comply with ISO 639 (2 levels).*
  - *Language name*
  - *The icon should be 16x11 px*
- *Click the* ***Create button****.*

> *More information about ISO 639.*

Edit a language

- *Click on* ***Edit language icon****.*
- ***Change the fields****.*
- *Click the* ***Edit button****.*

Delete a language

- *Click on* ***Delete language icon****.*
- ***Change the fields****.*

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

- *Click the **Delete button**.*

Edit language translations

At the top right, the translations can be filtered by key.

- *Click on **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 on the mobile view)*
- *extension (used on the frontend by extension modules)*

***Two key properties cannot be equal**.*

- ***Change the fields**.*
- *Click the **Edit button**.*

Export language translations

- *Click on **Export language translations icon**.*
- *The application **retrieves a SQL file** containing the language translations.*

Import language translations

Download your language package from Language translations.

- *At the bottom right **choose a language package file**.*
- *Click 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_7.1_ar-AR.sql***	*2147*
Azeri	***OpenKM_7.1_az-AZ.sql***	*1565*
Belgian/Flemish	***OpenKM_7.1_nl-BE.sql***	*911*
Bosnian	***OpenKM_7.1_bs-BS.sql***	*584*
Bulgarian	***OpenKM_7.1_bg-BG.sql***	*909*
Català	***OpenKM_7.1_ca-ES.sql***	*1199*
Chinese (Simplified)	***OpenKM_7.1_zh-CN.sql***	*1275*
Chinese (Traditional)	***OpenKM_7.1_zh-TW.sql***	*1279*
Czech	***OpenKM_7.1_cs-CZ.sql***	*2233*
Danish	***OpenKM_7.1_dk-DK.sql***	*884*
German	***OpenKM_7.1_de-DE.sql***	*2204*
German - Austria	***OpenKM_7.1_de-AT.sql***	*903*
Dutch	***OpenKM_7.1_nl-NL.sql***	*1334*
English	***OpenKM_7.1_en-GB.sql***	*1605*
English - USA	***OpenKM_7.1_en-US.sql***	*585*

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

Español	***OpenKM_7.1_es-ES.sql***	*1605*
Español Colombia	***OpenKM_7.1_co-ES.sql***	*584*
Euskera	***OpenKM_7.1_eu-ES.sql***	*1565*
Farsi	***OpenKM_7.1_fa-FA.sql***	*584*
French	***OpenKM_7.1_fr-FR.sql***	*1748*
Gallego	***OpenKM_7.1_gl-ES.sql***	*584*
Greek	***OpenKM_7.1_el-GR.sql***	*1205*
Hebrew	***OpenKM_7.1_he-IL.sql***	*2235*
Hungarian	***OpenKM_7.1_hu-HU.sql***	*2219*
Indonesian	***OpenKM_7.1_id-ID.sql***	*585*
Italian	***OpenKM_7.1_it-IT.sql***	*1415*
Japanese	***OpenKM_7.1_jp-JP.sql***	*890*
Korean	***OpenKM_7.1_ko-KR.sql***	*1902*
Latvian	***OpenKM_7.1_lv-LV.sql***	*585*
Lithuanian	***OpenKM_7.1_lt-LT.sql***	*584*
Macedonian	***OpenKM_7.1_mk-MK.sql***	*585*

Norwegian	***OpenKM_7.1_nb-NO.sql***	*1175*
Polish	***OpenKM_7.1_pl-PL.sql***	*1286*
Portuguese (Brazil)	***OpenKM_7.1_pt-BR.sql***	*2128*
Romanian	***OpenKM_7.1_ro-RO.sql***	*585*
Russian	***OpenKM_7.1_ru-RU.sql***	*1242*
Serbian	***OpenKM_7.1_sr-SR.sql***	*942*
Slovak	***OpenKM_7.1_sk-SK.sql***	*1447*
Suomi	***OpenKM_7.1_fi-FI.sql***	*1103*
Swedish	***OpenKM_7.1_sv-SE.sql***	*849*
Thai	***OpenKM_7.1_th-TH.sql***	*585*
Turkish	***OpenKM_7.1_tr-TR.sql***	*1259*
Vietnamese	***OpenKM_7.1_zh-VN.sql***	*1227*

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

How to create a new translation

Before starting a new translation, please check if it is available in the Language translations section.

A sample of a new language based on Oromo from Ethiopia

*The first step is identifying what your language code is based on ISO 639 (level 2). 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 will be 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** containing 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 to be used 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* • *Purge data* • *Purge trash*
Performance	• *Cache stats* • *Clean conversion cache* • *Hibernate stats* • *Profiling stats*
Low-level access	• *Database query* • *Scripting*
Check the specific configuration	• *Check email* • *Check text extraction* • *LDAP Search* • *System properties* • *Create support zip*
Import & export	• *Database import* • *Repository import* • *Repository export*
Search engine	• *List indexes* • *Rebuild indexes*

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

Others	• *Plugins* • *Repository checker*
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 for managing users and roles.

On this view, a table is shown with all users who have administrative access based on the admin role.

More information about the default admin role is available at Security configuration parameters.

Change user access level

- *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 retrieval of data.

> *For example, the users and roles list, when the application is configured to work with LDAP, are cached to reduce the 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*

Also, the cache can be used to collect data useful to identify 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:

Cache name	Hits	Misses	Objects	Memory hits	Memory objects	Disc hits	Disc misses	Disc objects	Action

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 in the cache disk.	true

Disc misses	*Number of failed queries in the cache disk.*	*true*
Disc objects	*Number of objects in cache disk.*	*false*
Action	*Clean cached objects.*	*Not applicable.*

Clean all cache

- *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 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****.*

Disables 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 on the **Check** button*

If the application mail configuration is set correctly, it will return "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 on the general application user interface).*
  - *Document path.*
  - *Document outside application (choose a document from the file system to be uploaded).*
- *List of available text extractors.*

Check text extraction

- ***Choose a document** to be processed*
  - *OpenKM document:*
    - *Fill the document UUID or document path field.*
  - *External document:*
    - *Choose a document from the file system to be uploaded.*
- ***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's 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.

File conversion cache is stored in the file system, by default in the $TOMCAT/repository/cache folder.

A common 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 to an image. This process consumes some CPU.

To prevent consuming more CPU than necessary, the documents are converted only when necessary: the first time or after a document update. 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 all of them.

Cached files:

Type	*Description*
DXF	*Caused by AutoCAD pre-visualization conversion.*
PDF	*Several causes:* • *Document pre-visualization.* • *User conversion on demand.*
SWF	*Document pre-visualization.*
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****.*

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.

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 the source OpenKM.*
Concurrent documents	*Check if you want to import this data.*
Concurrent folders	*Check if you want to import this data.*
Concurrent records	*Check if you want to import this data.*
Concurrent mails	*Check if you want to import this data.*
Nodes	*Check if you want to import this data.*
Users and roles	*Check if you want to import this data.*
Node relations	*Check if you want to import this data.*
Bookmarks	*Check if you want to import this data.*

Import

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

- *Set the DB URL.*
- *Set the DB user.*
- *Set the DB user password.*
- *Set the source folder.*
- *Set the destination folder.*
- *Set the thread pool size.*
- *Set the source tenant ID.*
- *Check the database import options.*
- *Click on* ***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 software client.

The results of the queries can be exported to SQL or CSV file formats.

Allowed types:

Type	*Description*	*List*	*Export*
JDBC	*SQL queries supported by the configured database.*	*When chosen, a list of available tables is shown.*	*Yes.*
Hibernate	*Neutral database queries based on the 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 a 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 on 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 on the **Execute button**.*

> *When "Show SQL" option is checked, the results show the real SQL executed.*

Example:

```
select count(*) from NodeBase;
select count(*) from NodeFolder;
```

Allowed sentences

- *On the bottom right **choose a valid SQL file**.*
- *Click on the **Import SQL script button**.*

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, identifying performance problems and their 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 are suffixed 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.

*In Query statistics, all the database queries executed by the application are shown. The most important column is **Max dur.**, which indicates the maximum duration of the query.*

Enable Hibernate stats

This option is only available when Hibernate stats is stopped.

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

- *On the top right click on* *Enable button.*

Clear Hibernate stats

- *On the top right click on* *Clear button.*

Disable Hibernate stats

> *This option is only available when Hibernate stats is running.*

- *On 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 the LDAP configuration.

> *The application's LDAP integration is based on a subset of LDAP queries, for example, retrieving LDAP users or LDAP groups.*
>
> *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*

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 in the Lucene search engine.

On the main screen, a table is shown with the Lucene-indexed fields of each object.

Common indexed fields:

Field	*Description*
#	*Number*
_hibernate_class	*The Hibernate class name:* • *com.openkm.dao.bean.NodeFolder* • *com.openkm.dao.bean.NodeDocument* • *com.openkm.dao.bean.NodeMail* • *com.openkm.dao.bean.NodeRecord*
uuid	*Object'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 may be additional indexed fields.*

Search indexes

- *On the top right, click on* ***Search indexes button****.*
- *In the input box, copy an object UUID (from the desktop view you can get any object's UUID at 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 by this search:

```
name:paran*
```

But be careful, because using quotation marks you won't get any results:

```
name:"paran*"
```

If you want to locate all PDF documents whose name starts 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 name starts with "parana" and 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, an extra field named terms is 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****.*

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 clean all Tomcat logs.

- *On the top right, click on **Purge button**.*

Download file as ZIP

- *Choose a log file.*
- *Click on **download as ZIP icon**.*

Browse log file

- *Choose a log file.*
- *Click on **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 some text.*

Plugins

The application comes with a plugin feature system that offers an easy way to customize and extend OpenKM functionalities. By default, the application includes a general-purpose plugin implementation; these can be disabled, replaced by others, or increased in number depending on the plugin type. For example, for the antivirus plugin it makes sense to disable it or replace it with another; in the case of the text extractor, disable it or increase the number of available text extractors.

Plugins are shown in a table structure:

Plugin family name		
Name	***Class name***	***Status***

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*
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*

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

		• *Creating your own Option Select Values plugin*
Field validators	*Metadata*	*Validation is based on server-side logic. It helps implement complex validation logic that is not covered by 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 directory 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*
Document converters	*Conversion*	*The conversion service is used to convert a document to other formats.* *More information at:* • *Creating your own Document Converter*
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:*

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

		• *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 provides low-level, valuable information to the administrator to help identify 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 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.

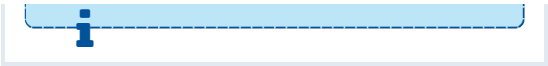
Profiling stats are shown in a table structure:

Class	Method	Calls	Max dur.	Min dur.	Avg dur.	Total dur.	Action

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.*
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

- *In the top right, click on* ***the Enable button****.*

Disable profiling stats

- *In the top right, click on* ***the Disable button****.*

Clear profiling stats

- *In the top right, click on* ***the Clear button****.*

Refresh profiling stats

- *In the top right, click on* ***the 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 Subscription Users.*

Purge

This operation cannot be reverted; use with care.

- *In any list, click on* ***Delete icon****.*

Filter

- *To search in any list, click on* ***Filter button****.*

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

Purge trash

The purge trash feature helps the administrator clean all users' trash.

When a user deletes objects (folders, documents, emails, or records), they go into 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. This can lead to a lot of garbage in the trash.

The administrator should periodically check the repository's growth, and especially the size of the trash.

Purge

This operation cannot be reverted. Use with care.

- ***Write 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	*Rebuilds indexes with actual contents.* *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 contents that no longer exist 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 due to a hard disk failure or something 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	*Fast verification will only check for the existence of the last 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 size and the repository options checked.

The repository checker process will show a progress status as a percentage. Before it is shown, the application calculates the total number of nodes that should be checked; in a large repository this calculation can take several minutes. Please be patient.

- *Set the path.*
- *Choose the repository checker options (fast, versions, checksums).*
- *Click on 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 to export the whole or part of the OpenKM repository to the file system.

The output will be the same folder and file structure that is 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

If metadata and history are not checked

The result will be a folder structure containing documents and mails. Although a document has several versions, only the

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

current version will be exported.

If only metadata is checked

The result will be a folder structure containing documents and mails. Although a document has 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}
}
```

Take into consideration that the JSON output format can change if the main objects (folder, document, mail, or record) are added, removed, or modified.

If 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)*

If 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, a ***.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 the application is integrated with third-party software, the third-party software usually stores the node's UUID in its own database as the primary reference. These references must 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 was an advanced feature that enables administrators to execute some BeanShell scripts in folders, triggered on every notified event (for example, uploading documents).

This feature has been enhanced by Automation (see ExecuteScripting at Built-in Automation Action plugins), which helps you 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 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) each time the OpenKM demo is restarted.*

Execute a script as part of 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.)*

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.

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.

User guide

You can use the Quick start guide to learn the basics quickly and to find detailed information about the user sections.

The OpenKM front is the back-office view of the content repository. The features are organized into several sections:

- *The **Menu bar** at the top is available for all the features organized by type.*
- *The **Quick search box** is at the top right.*
- *The **Toolbar** contains the most commonly used features.*
- *The **Desktop** tab enables you to navigate the repository.*
- *The **Search** tab enables a query of the repository.*
- *The **Dashboard** tab provides a set of dashboards.*
- *The **Administration** tab provides a set of administration tools.*
- *The **Status** panel at the bottom provides general information.*

*Only members with the "**ROLE_ADMIN**" role have access to the **Administration** tab.*

The OpenKM front allows you to manage any content: scanned images, photographs, office documents, and even large video files or CAD files, among others. It also enables social activities around content to connect people with other people, information with information, and people with information, helping to manage the company's collective intelligence more efficiently.

The OpenKM Profiles feature helps administrators to 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 control of document versions.*
- *Reduce the time and resources used to locate information.*
- *Avoid knowledge leaks or wrongful access by users, thereby complying with the obligations set out in the LOPD (Spanish Data Protection Law).*
- *Restrict unauthorized users' access to information.*

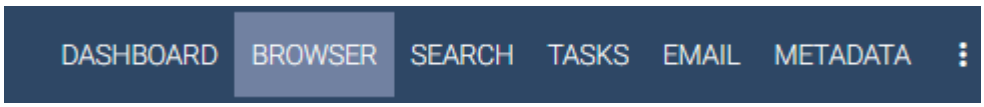
Through the OpenKM system, all relevant and valuable information can be accessed and found more quickly to resolve problems and make decisions.

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

Menu bar

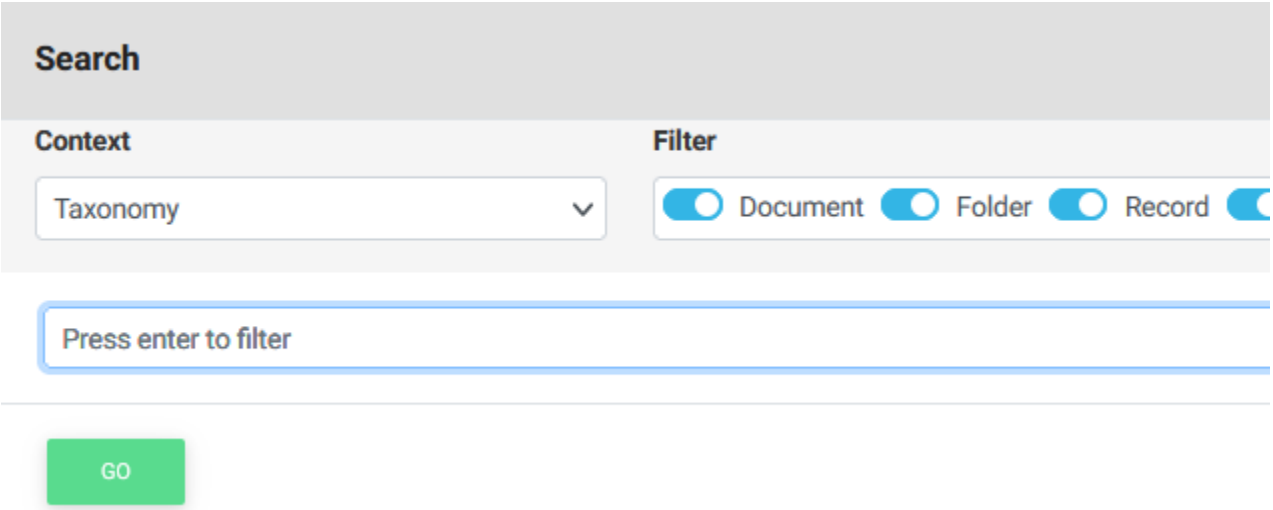
The menu bar at the top has all features organized by type:

- *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.

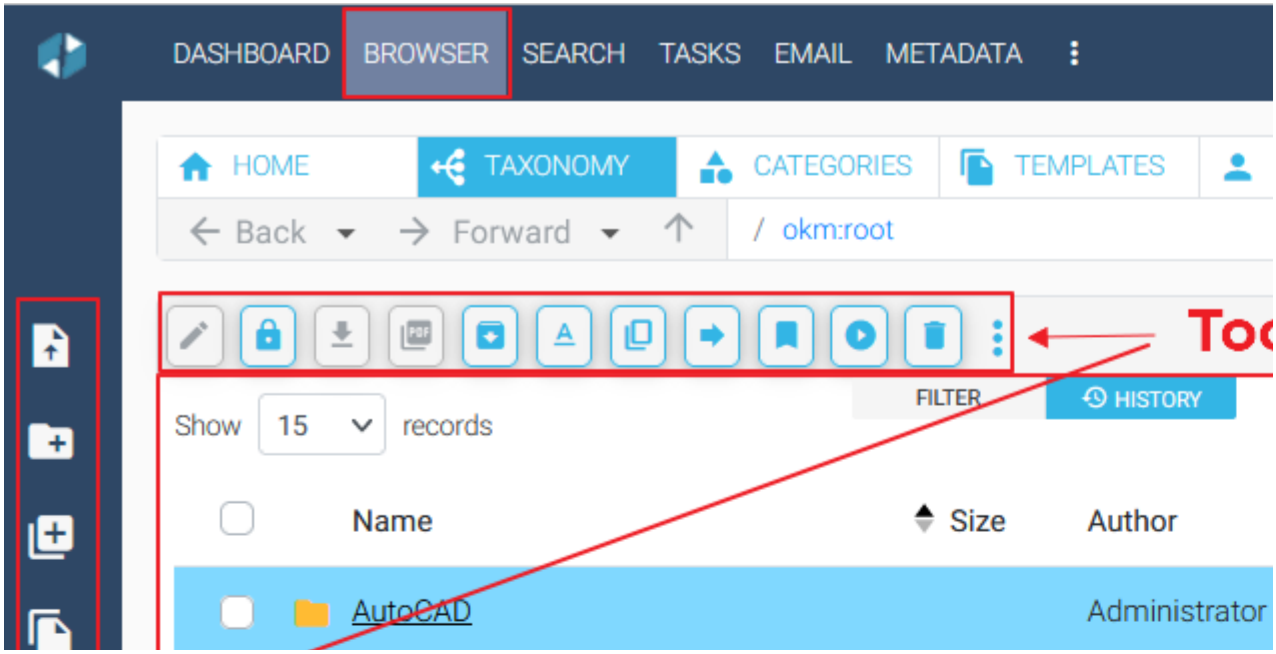
Browser

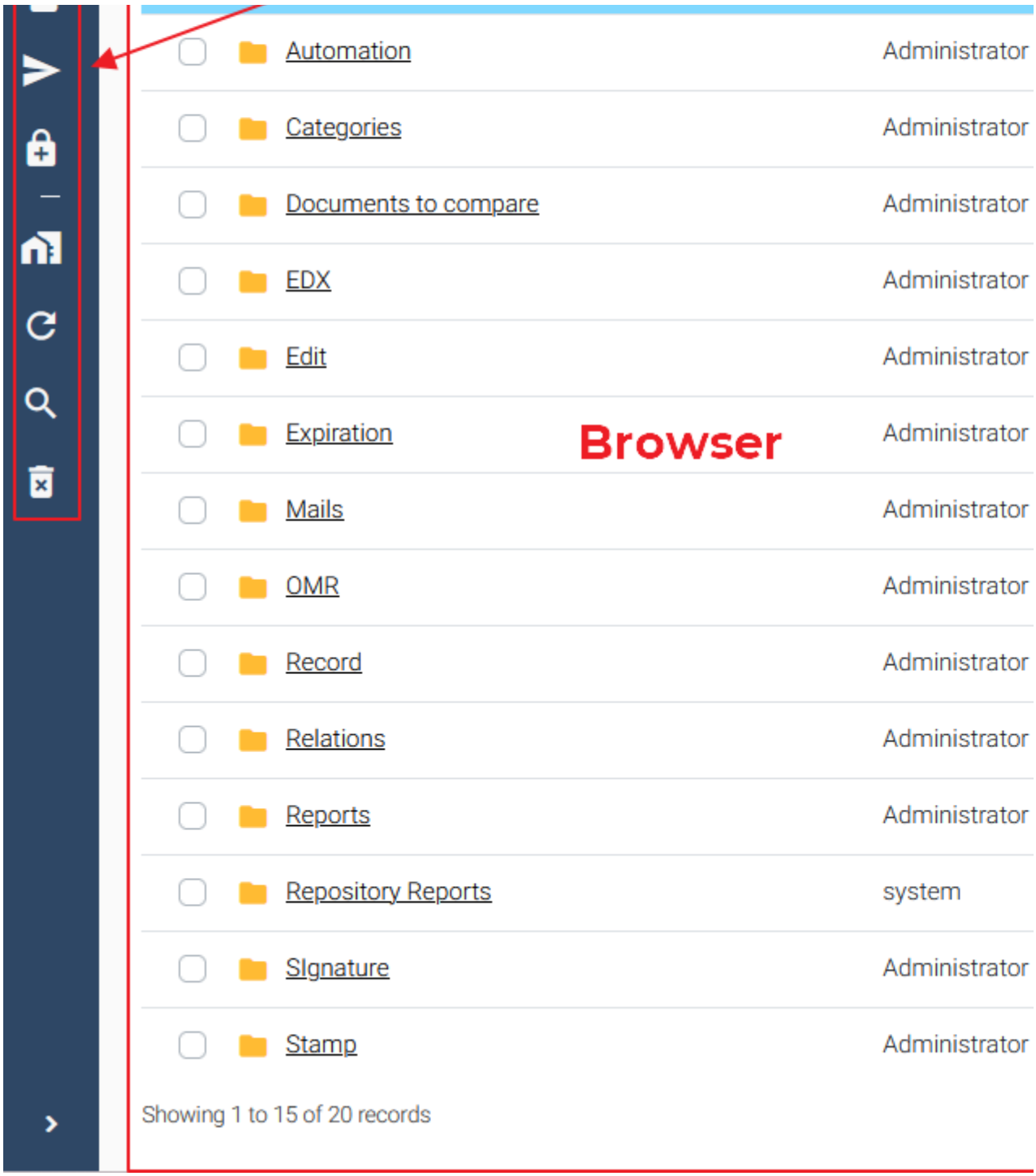
The browser panel at the top center shows a table of nodes (documents, folders, emails, and records) for the selected navigation node.

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 navigator node or a browser node.

> *The selected node in the navigator panel determines the content shown in the browser panel. If the last node chosen 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.*



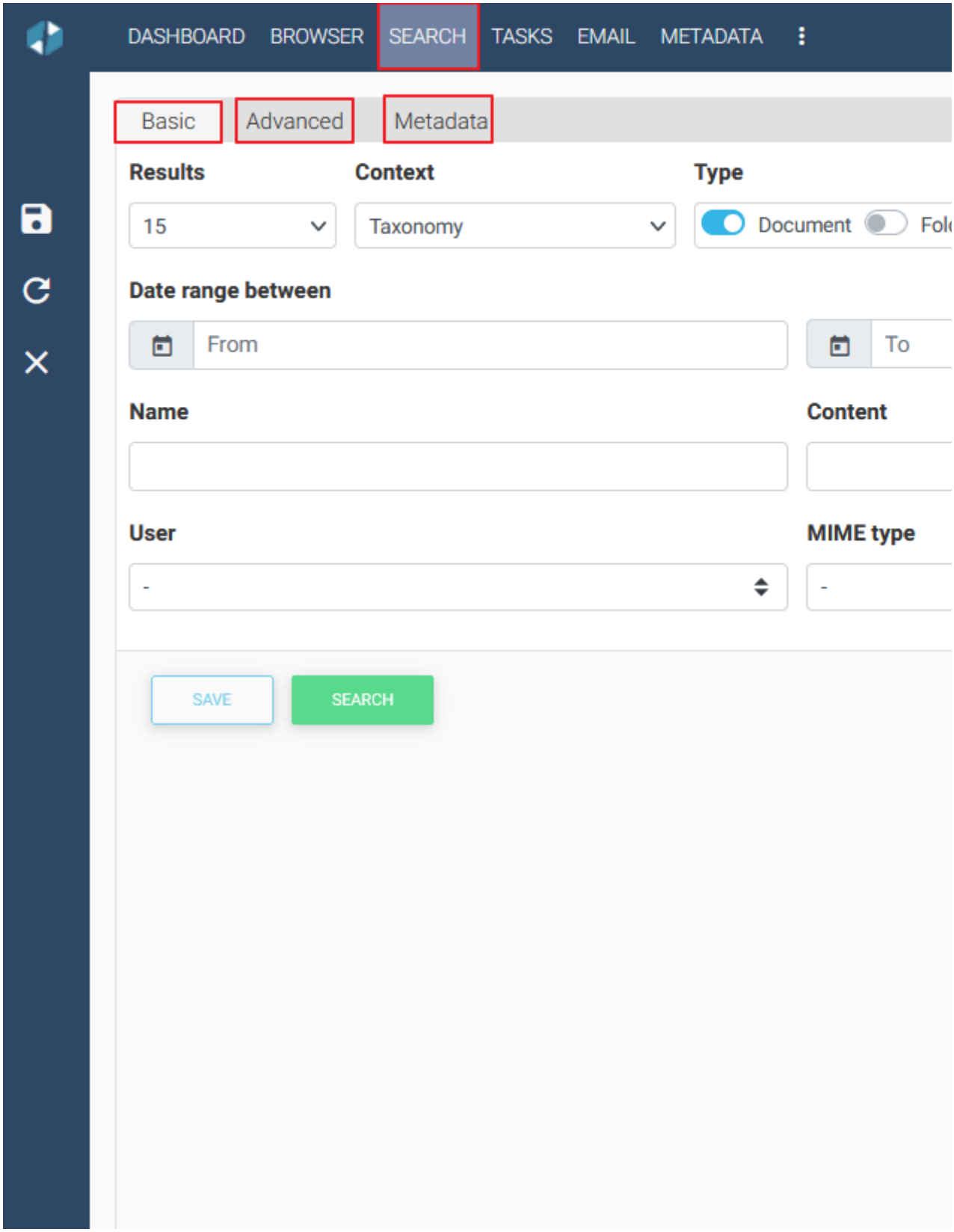


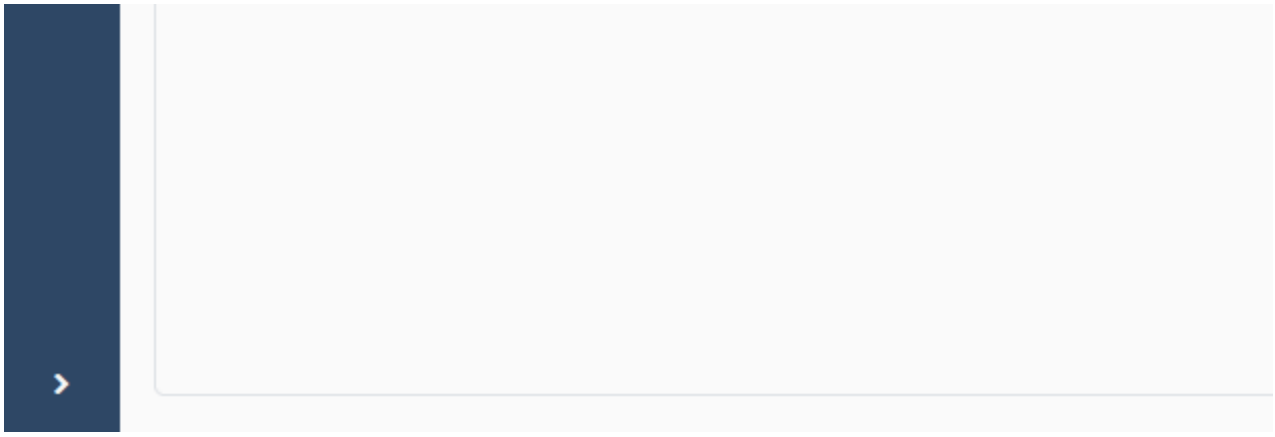
Search

The Search tab enables querying the repository. This tab gives access to different search options to query the content of the 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.*



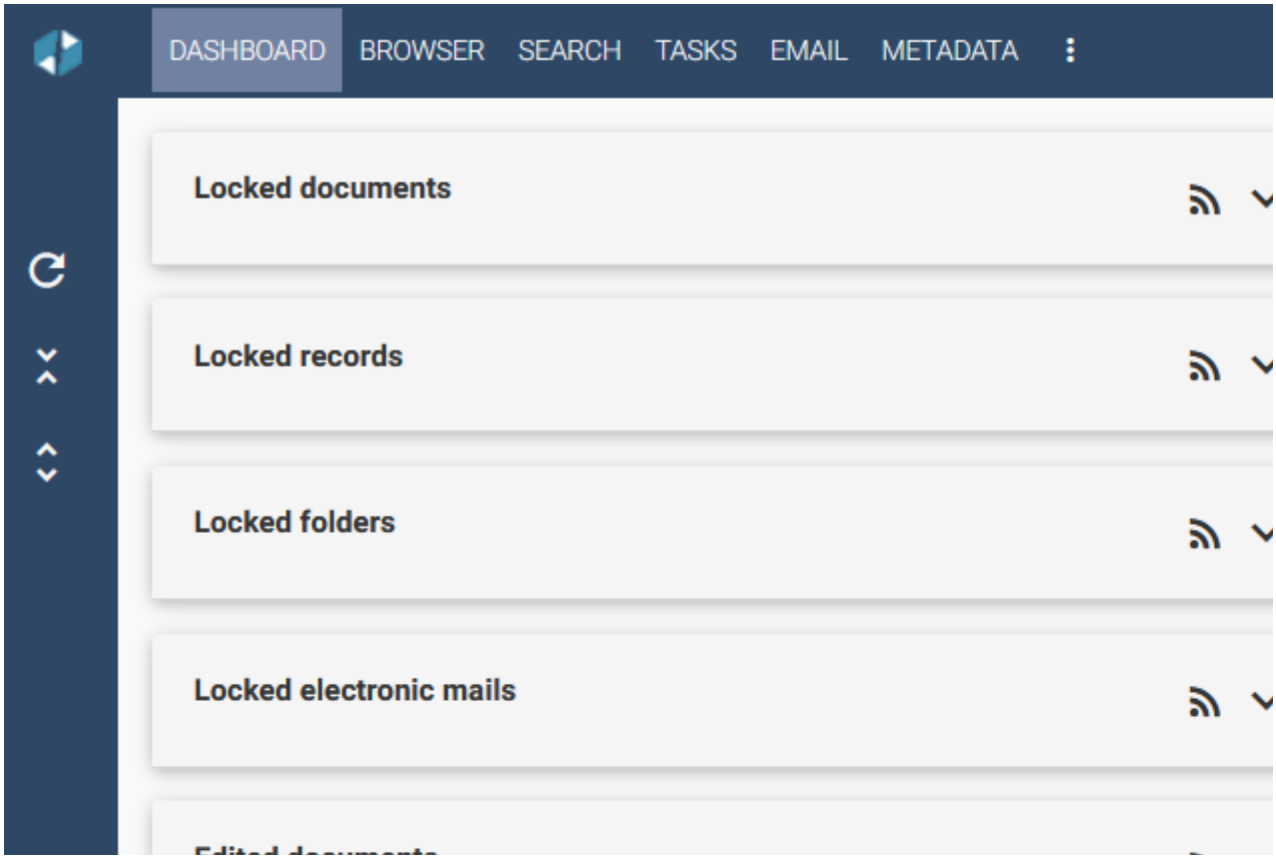


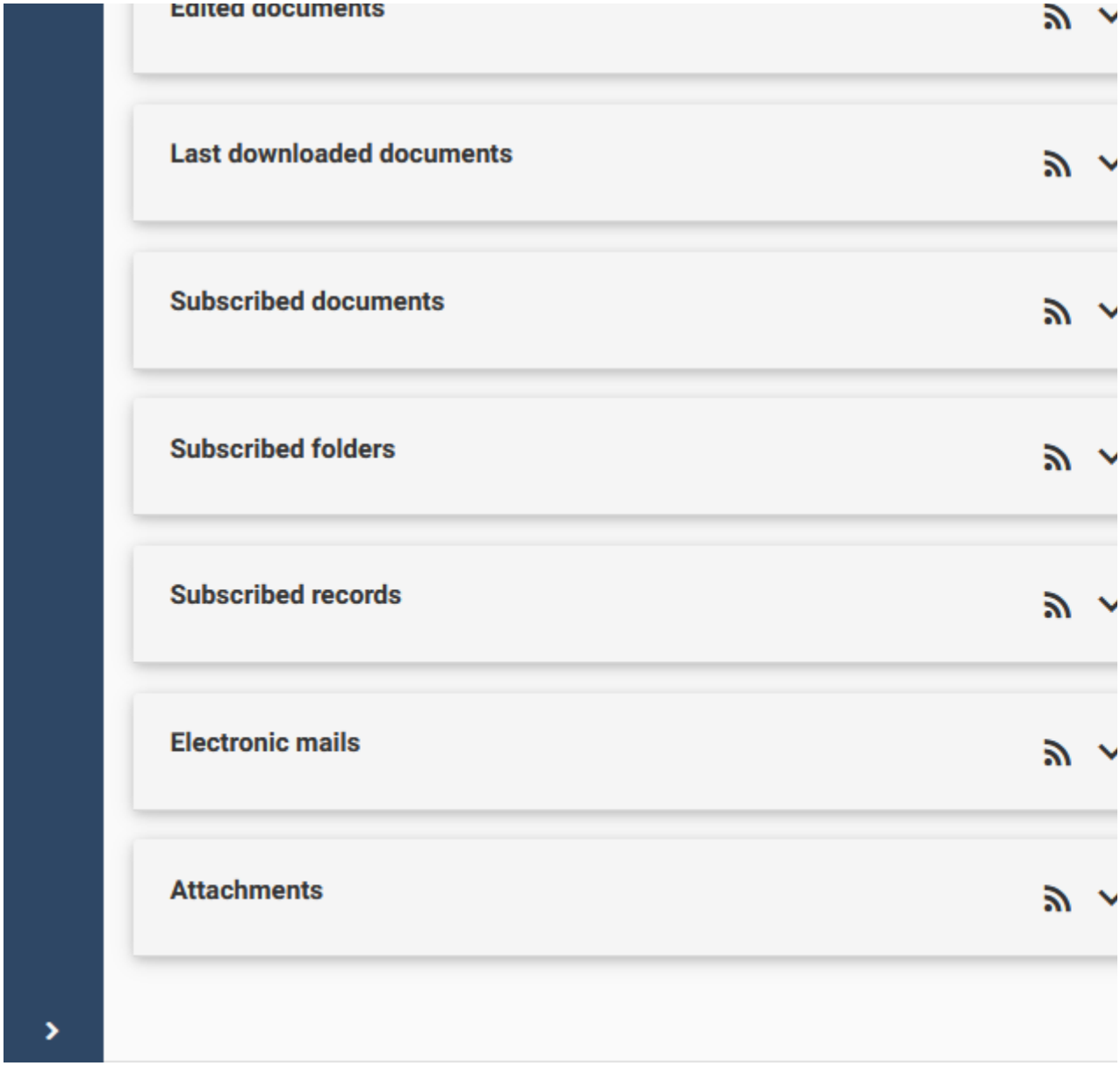
Dashboard

The dashboard panel provides a set of dashboards; each user has a set of personal dashboards that display 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.*





Administration

The administration panel is a set of tools that help the administrator manage the application.

> *More information is available 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 edition.*
- *Message panel.*

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 the features of OpenKM:

- *Logging in.*
- *Create a taxonomy.*
- *Add documents.*
- *Edit documents.*
- *Search documents.*

Logging in

You need to log in to OpenKM.

- *Enter the following on the login page:*
  - ***Username**: your username.*
  - ***Password**: your password.*
- *Choose your **language**.*
- *Click on the **Login** button.*

Ask your administrator for your login URL. By default, the OpenKM server URL is http://127.0.0.1:8080/openkm

The default administrator credentials are:

- ***Username**: okmAdmin*
- ***Password**: admin*

*Do not forget to **change the administrator password**.*

Create a taxonomy

The dictionary says:

"Taxonomy (from Greek taxis meaning arrangement or division and nomos meaning law) is the

science of classification according to a pre-determined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval. In theory, the development of a good taxonomy takes into account the importance of separating elements of a group (taxon) into subgroups (taxa) that are mutually exclusive, unambiguous, and taken together, include all possibilities."

*A **taxonomy** is a way to group things. Like a controlled vocabulary focuses on concepts, in practice, a good taxonomy should be simple, easy to remember, and easy to use.*

The first step should be analyzing the company organization and defining a folder hierarchy to store the organized documents.

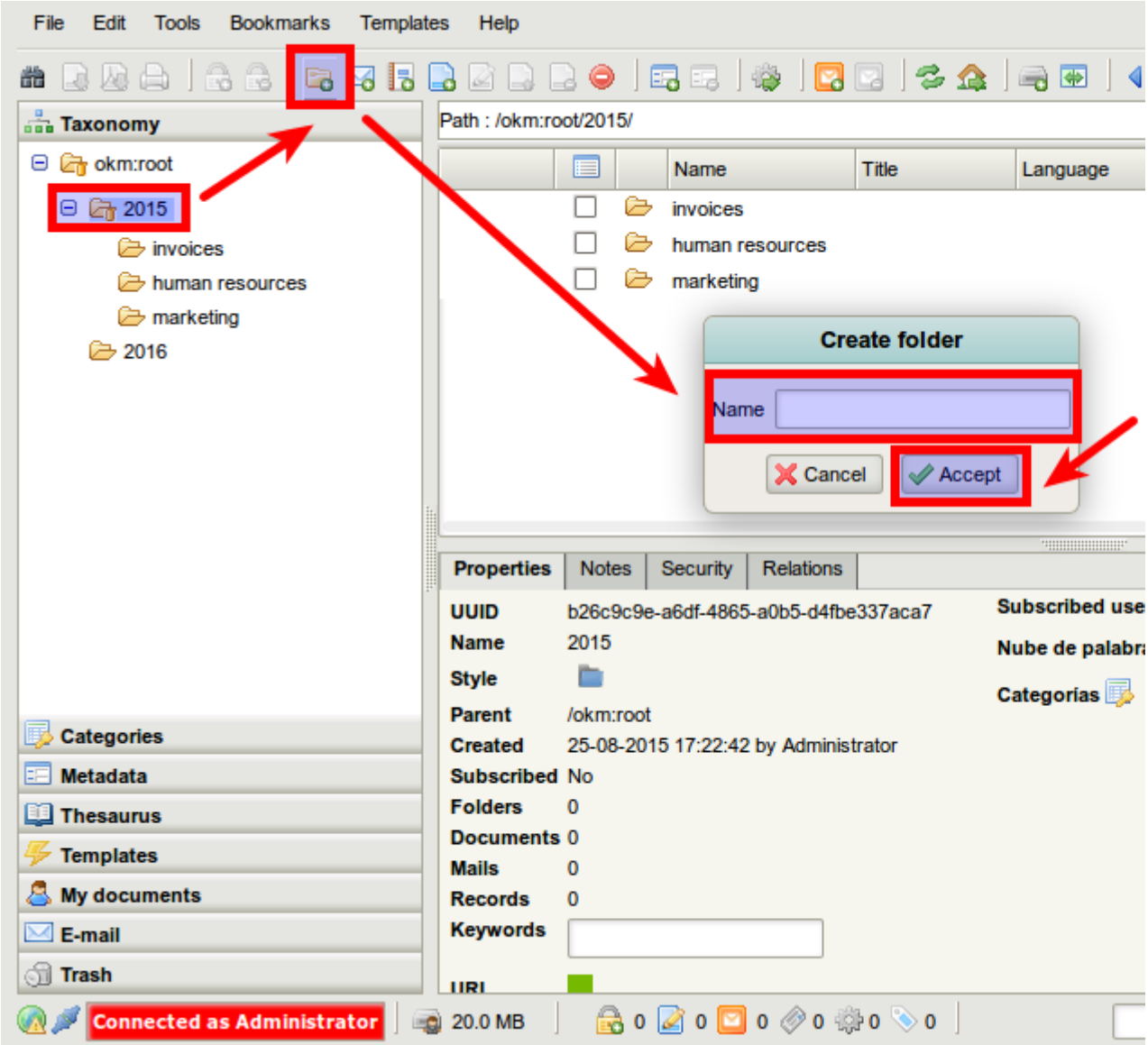
Take this one as a sample:

/okm:root/	***2015/***	***invoices/***
		human resources/
		marketing/
	2016/	***invoices/***
		human resources/
		marketing/

To create a folder:

- ***Select a node** from the taxonomy where you want to create the folder.*
- *Click on the **folder icon** in the toolbar.*
- *Set the **name**.*
- *Click the **Accept** button.*

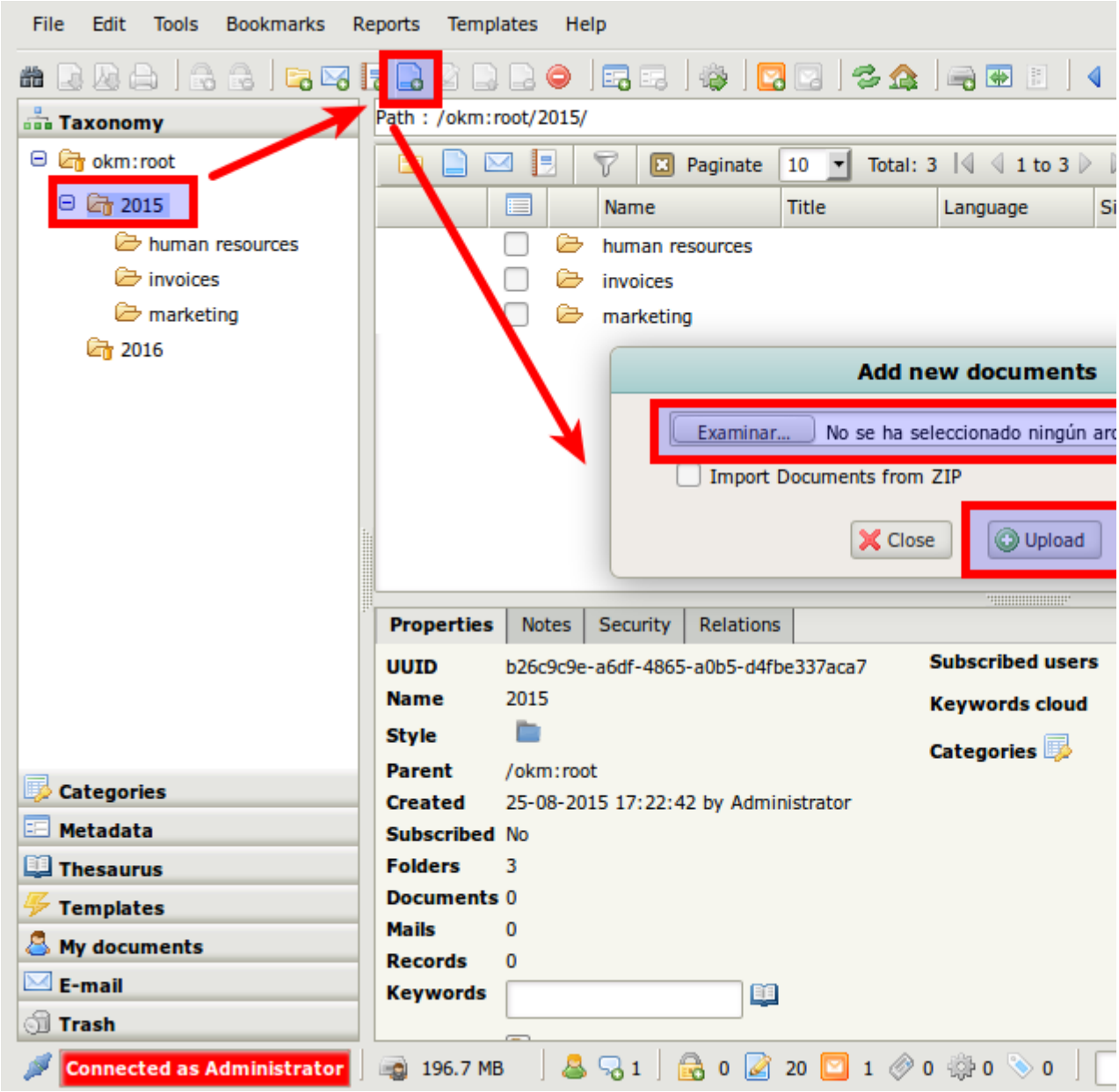
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



We suggest spending some time thinking about and designing your taxonomy. We suggest less than 200 elements per folder for the best UI performance.

Add documents

- ***Select a node*** *from the taxonomy where you want to add the document.*
- *Click the* ***add document*** *icon in the toolbar.*
- ***Select the document*** *from your computer.*
- *Click the* ***Upload*** *button.*



Edit documents

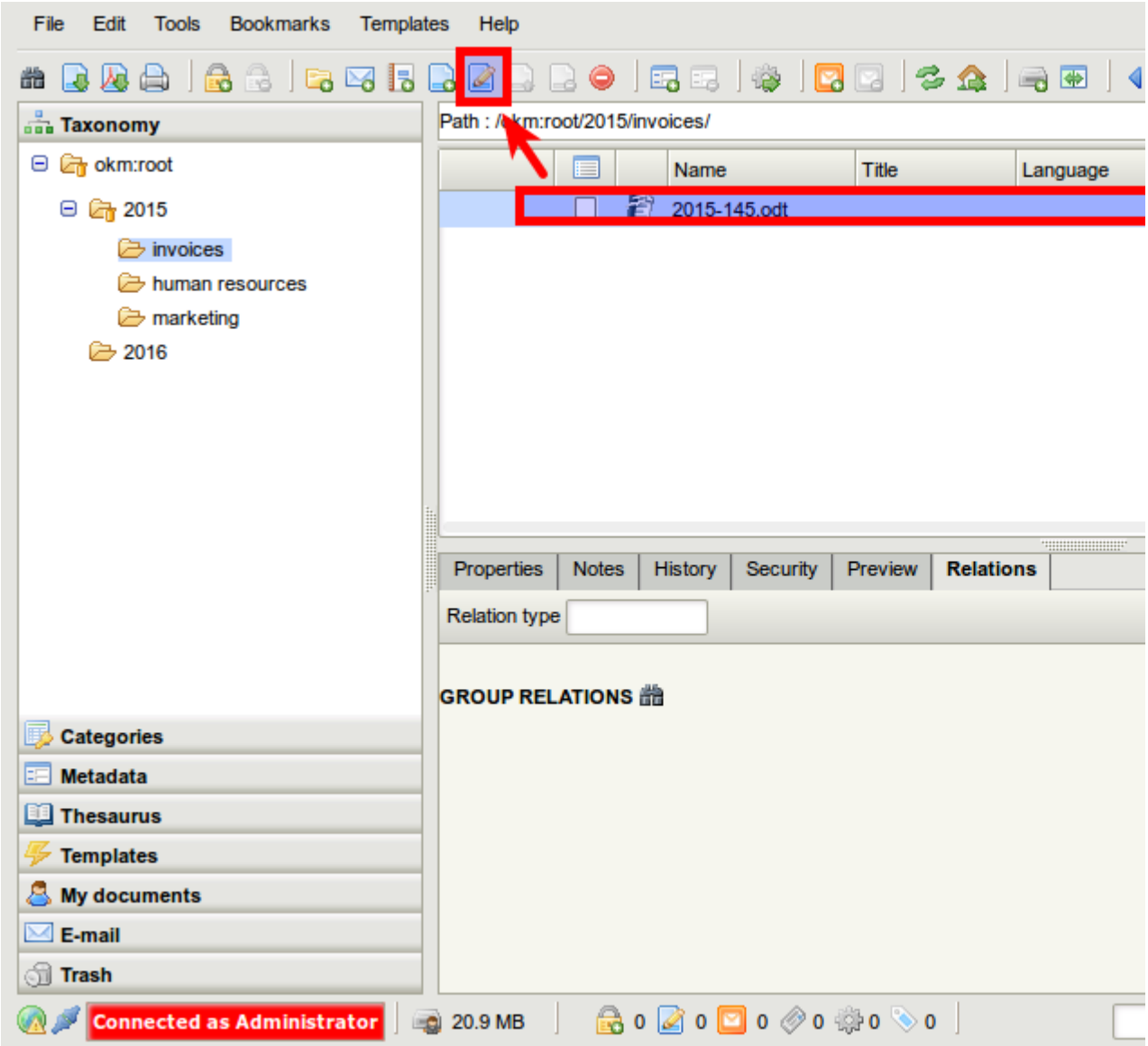
*Document editing is based on the "**check-out / check-in**" model.*

> *"**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 want to simultaneously 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.*

When a document is checked out to a user, that user can work on that document, and other users cannot. Other users must wait for the user to check the document before they can edit. While the document is checked out, other users can see only the last version that was checked in; they can't see the changes the current user has made to the document while it's been checked out.

First, check out the document:

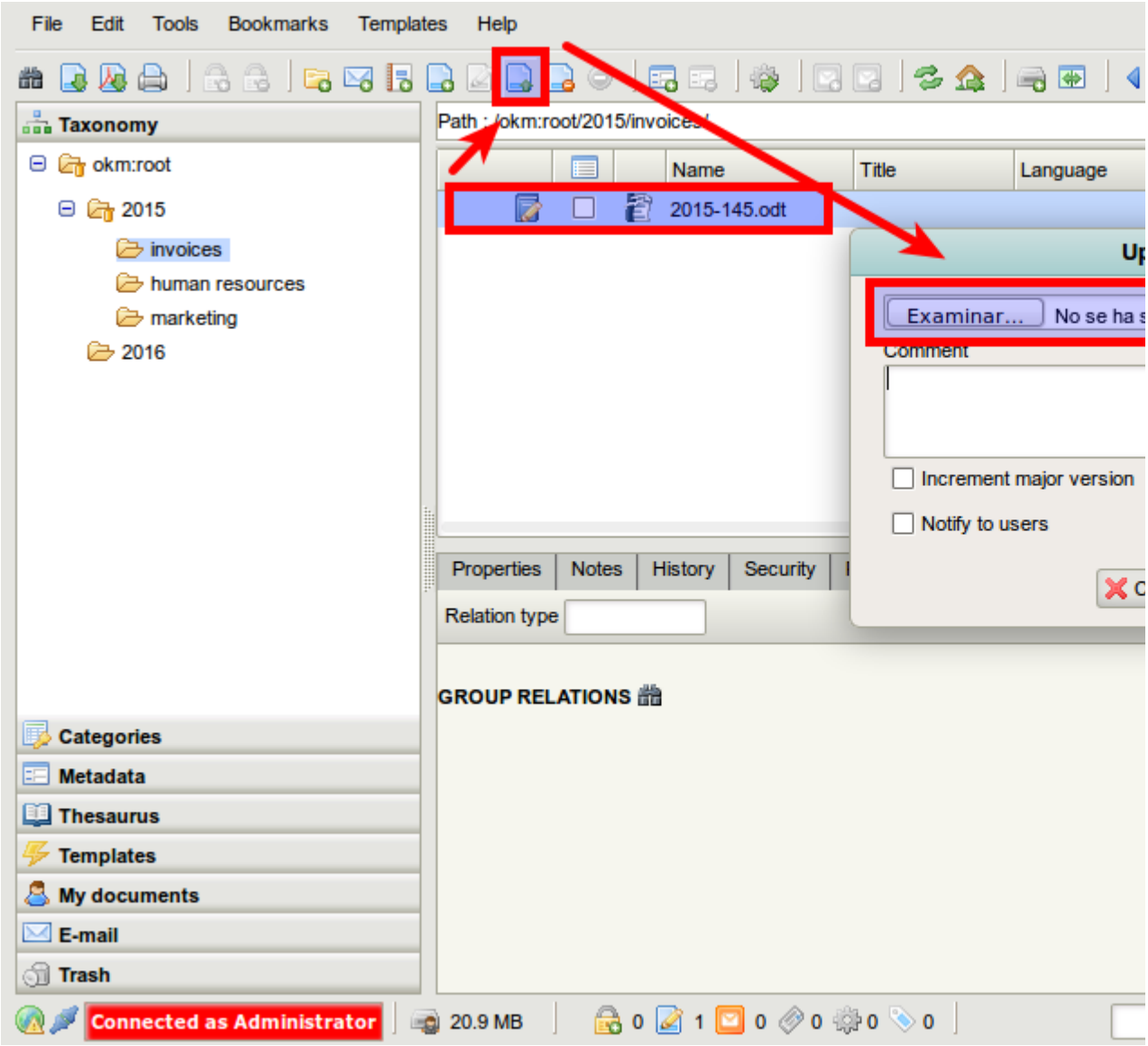
- ***Select a document***.
- *Click on the* ***Edit icon***.



Finally, check in the document to upload the changes:

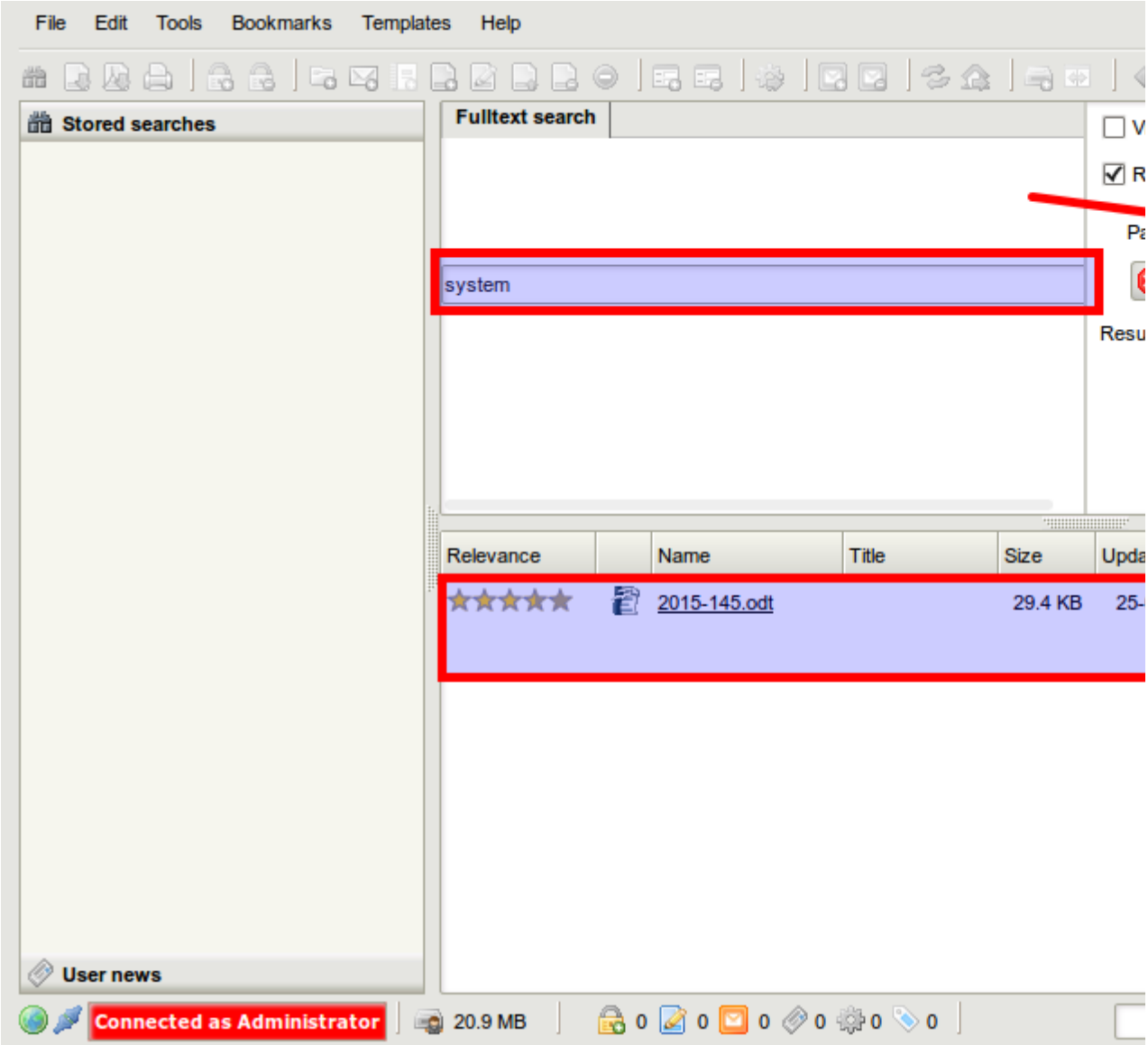
- *Select the* ***edited document***.
- *Click on the* ***Update icon***.

- ***Select the document*** *from your computer.*
- *Click the* ***Upload*** *button.*



Search documents

- *Go to* ***search view****.*
- ***Enter the search term*** *for the query.*
- *Click on the* ***Search*** *button.*


File
Edit
Tools
Bookmarks
Templates
Help
Stored searches
Fulltext search
system
Relevance
Name
Title
Size
2015-145.odt
29.4 KB
User news
Connected as Administrator
20.9 MB

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 it is not a file, and 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 (in the mail context, these are called attachments).*

*In essence there are basically two kinds of nodes of type **record**:*

- *Non-digital records.*
- *Digital records.*

*A **non-digital record** is only a set of metadata groups.*

*A **digital record** can 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 categorizing nodes is good practice. A good categorization will help users find the desired information quickly, using less time, saving money, and being more efficient in your business processes.

OpenKM helps 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, I n practice, a good taxonomy should be simple, easy to remember, and easy to use.*

For best performance in the UI we suggest - when possible - 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 contents. 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 keywords usage:

- *When you have thousands of keywords, searching by keyword can be inefficient (you are not able to refine the search and you get a lot of results with the same keywords because users do not follow a restricted way to catalog). 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 keyword dictionary.*
- *Users write keywords in a similar way but not always the same (for example, users use the words "building", "construction", and "edifice" interchangeably).*

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

You can get the best results when:

- *You limit the number of users who are able to set keywords.*
- *You use a controlled keywords dictionary (more information at http://www.w3.org/2004/02/skos/).*
- *You share a complete company keywords dictionary with your users. That helps users understand how knowledge is mapped in your organization and helps them find it.*

Categories

In OpenKM, the categories are a virtual taxonomy that help with navigation of 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 a certain node's content. Metadata groups help create controlled classifications and are a powerful way to locate nodes and refine query searches.

With metadata groups you can simulate the keywords and categories features. We encourage spending 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 of 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 offers 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 located?*
- *What kinds of documents do you have?*
- *Are your documents in electronic format or not?*
- *What document formats do you have? PDF, MS Office, OpenOffice, etc.*
- *How are you currently finding documents? How would you like to find them; 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) which information is relevant (metadata). For example, invoices usually have an invoice number, a date range, and a customer, among other fields. The metadata will help users retrieve information quickly and efficiently.*
- *Identify the security that must be applied for each document type: who can read, write, or delete information. Also, your security needs will help you 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 with the implementation of the application? Usually they could be:

- *Set some order in my organization. The application enforces users to work in certain ways, which benefits your company because it helps you set rules that all users must follow.*
- *Consolidate the intellectual capital. By consolidating all your company's knowledge in a single place, you make that knowledge available to be shared among all your users, with rules applied to it.*
- *Help you 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 practice

- *Consider OpenKM as a project that affects all your departments and usually your customers.*
- *Choose the leaders.*
- *Gradually implement the project. You can start with a single 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 receive any news in the company as something negative; however, if they perceive the application as helping their work (efficient search, saving time, and solving real problems, etc.), they will be the first to be interested in using it. One of the major goals of OpenKM is the intuitive UI, which helps in this process (after two weeks the user feels comfortable using the application).*
- *Think simply; the best solution is usually the easiest way to solve problems.*
- *Listen your users and try to get them involved in the project.*
- *Periodically review your plan, try to refine and identify possible problems.*

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

Working with the menu bar

On the menu bar at the top, 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. Your menu bar may show options that are 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 the browser panel to apply the same action.

File

Feature	*Description*	*Massive*
Find folder	*Finds the 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 a mail in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*

Find record	*Finds a record in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*
Create folder	*Creates a new folder in the selected node of the navigator panel.*	*No.*
Add document	*Uploads a new document into the selected node of the navigator panel.* *When the option "Import documents from ZIP" is checked, you import the contents of a ZIP file, respecting the hierarchy of the folder tree. If you have problems with filename encodings, please take a look at Client configuration. You can also upload RAR or JAR MIME type files.* *When the option "Notify 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 can 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 in the browser panel.*	*No.*
Download document as PDF	*Downloads the selected document in the browser panel in PDF format.* *The feature is enabled when it is possible to convert the document to PDF format.*	*No.*
Send link	*Sends a URL to the selected document in the browser panel by email.*	*Yes.*
Send	*Sends the selected document in the browser panel as an attachment by mail.*	*Yes.*

document attachment		
Create from template	*Creates a new document based on an existing one in the templates space.*	*No.*
Export to ZIP	*Downloads the selected nodes and compresses them into a ZIP file. If you have problems with filename encodings, please take a look at Client configuration.*	*Yes.*
Refresh	*Refreshes the user interface.*	*No.*
Restore	*Restores the selected node from the trash bin.*	*No.*
Purge	*Purges the selected node from the trash bin.*	*No.*
Purge trash	*Cleans the user's trash bin.*	*No.*
Exit	*Exits the application.* *Users should always exit the application using this menu option and not just close the web browser. In any case, after 30 minutes the user's web session on the server will be automatically terminated.*	*No.*

Example 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 an asterisk "**?**" 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 application, always add "*****" at the end.*

Edit

Feature	*Description*	*Massive*
Lock	*Locks the selected document in the browser panel.*	*Yes.*
Unlock	*Unlocks the selected document in the browser panel.* *Only the user who locked the document or a user with administrative 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 document editing.* *Only the user who edited the document or a user with administrative credentials (force cancel edit) can cancel the editing 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 their 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 form 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.*	*Yes.*
Add metadata group	*Adds a metadata group to the selected node.*	*Yes.*
Update metadata group	*Updates the values of the metadata group fields of the selected node.*	*Yes.*
Remove metadata group	*Deletes the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribes to the selected node.* *You can subscribe to any node type (document, folder, mail, or record). If you subscribe to a folder or record node, you'll be notified by email of any change to that node and its child hierarchy. Users will receive notifications based on the event definitions in their own profile (they will only receive notifications for events enabled in their profile). More information about configuring notification events on Profiles.*	*No.*
Remove subscription	*Unsubscribes from the selected node.*	*No.*

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

*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 to the document while it has been checked out.

Tools

Feature	Description	Massive
Languages	*Changes the user interface language.*	*No.*
Show administration	*Shows the administration tab on a separate browser tab.* *This option is only available for administrators.*	*No.*
User configuration	*Shows the user configuration. You can change your password and email address.*	*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.*

Bookmark

Feature	Description	Massive
Home	*Go to the home node.*	*No.*

Set default home	*Change the default home for the selected node.* *When a user logs in, the application opens in desktop view to the user's home node.* *By default, the home node is "**/okm:root**".* *You can select nodes in the navigation and browser panels from the desktop.* *More information at Working with desktop.*	*No.*
Add Bookmark	*Bookmark the selected node.* *You can select nodes in navigation and browser panels from the 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 the bookmark by the user.

When you click on a bookmark node, the application opens the node in desktop view.

Templates

*The template menu dynamically builds a menu hierarchy based on "**/okm:templates**" nodes. When you click on any menu option in Templates, the process of creating a new node begins (usually a new document based on a template document).*

Help

Feature	*Description*
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 that 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	*Finds a document in the repository.* *The search is executed only in the context of the selected stack.*	*No.*
Find mail	*Find mail in the repository.* *The search is executed only in the context of the selected stack.*	*No.*
Find record	*Find a record in the repository.* *The search can be executed only in the context of the selected stack.*	*No.*
Create	*Creates a new folder in the selected node of the navigator panel.*	*No.*

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

folder		
Download document	*Downloads the selected document in a browser panel.*	*No.*
Download document as PDF	*Downloads the selected document in a browser panel in PDF format.* *The feature is enabled when it is possible to convert the document to PDF format.*	*No.*
Lock	*Locks the selected document in a browser panel.*	*Yes.*
Unlock	*Unlocks the selected document in a browser panel.*	*Yes.*
Add document	*Uploads a new document into the selected node of the navigator panel.* *When the option "Import documents from ZIP" is checked, you import the contents of a ZIP file, respecting the hierarchy of the folder tree. If you have problems with filename encodings, please take a look at Client configuration.* *When the option "Notify users" is checked, you can send a mail notification to other application users.* *Optionally, based on **Administration** > Profiles configuration or **Administration** > Automation, a wizard is shown. The wizard can require you to set metadata groups, categories, or keywords.*	*No.*
Edit	*Edits a document (check-out).*	*Yes.*
Update	*Uploads a new document version (check-in).*	*No.*
Delete	*Deletes the selected node.*	*Yes.*
Cancel edit	*Cancels document editing.*	*Yes.*

Add metadata group	*Adds a metadata group to the selected node.*	*Yes.*
Delete metadata group	*Deletes the metadata group of the selected node.*	*Yes.*
Add subscription	*Subscribes to the selected node.* *You can subscribe to any node type (document, folder, mail, or record). If you are subscribed to a particular node of type folder or record, you'll be notified by mail when any change to that node or its child hierarchy occurs. Keep in mind that, in order to receive these mail notifications, you have to configure which events you want to be notified of in the user profile. See Profile Notification tab for more information.*	*No.*
Remove subscription	*Unsubscribes from the selected node.*	*No.*
Refresh	*Refreshes the user interface.*	*No.*
Home	*Goes to the home node.*	*No.*
Previous node	*Goes to the last node visited.*	*No.*
History	*Manages the history of nodes visited.*	*No.*
Next node	*Goes to the next node visited.*	*No.*

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

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 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, which means it is where multiple users can modify and share company data.*

*The **categories** stack is a virtual taxonomy space that helps with navigation of taxonomy nodes across other concepts.*

> *Categories are a hierarchical tree structure of category and subcategory folders. The idea is that 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 the metadata field values.*

> *The metadata view is a hierarchical tree structure of metadata groups. Each possible metadata value is represented as a tree node structure.*

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

Only fields of type Select, SuggestBox, and Checkbox are displayed (free-text fields like Inputs or Text area are not shown).

You can create your own Stack Metadata Adapter visualization. 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 multiple users to create new company documents.*

*The **My documents** stack is a private space for each user, which means that only the user can access this section and it is the default location where personal data is stored.*

*The **Email** stack is a private space for each user, which 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, which means that only the user can access this section. When a user deletes a node, it goes automatically to their own 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 Document", "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 (the selected node in the tree).

In the case of virtual stacks (categories, metadata), the nodes shown in the browser panel do not have the same parent (the selected node in the tree); they are shown because they have a relation (category or metadata value) with the selected node in the tree.

*When you do a **right click** with the mouse button on a node in the tree, a **contextual menu** appears. 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 a folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Add document.*

- *Add a record.*
- *Add a bookmark.*
- *Set the default home.*
- *Export to ZIP.*

*Available menu options in the **Categories** stack are:*

- *Create a folder.*
- *Delete.*
- *Move.*

*Available menu options in the **Templates** stack are:*

- *Create a folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options in **My documents** stack are:*

- *Create a folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Export to ZIP.*

*Available menu options in the **E-mail** stack are:*

- *Create a folder.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy*

*Available menu options in the **Trash** stack are:*

- *Restore.*
- *Purge.*
- *Purge trash.*

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

Metadata view has no menu options.

Those folders to which the user is subscribed will be displayed in a different color to distinguish 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, mails, and records) that are in the selected node of the navigation panel.

*At the top of the browser panel, there's a **path** that indicates the actual location in the navigation panel (for example, "/**okm:root**/"). Below, there is a list of the nodes related to the selected node in the navigation 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**).*
- *Checkbox (to apply actions on several nodes at the same time).*
- *Icon (indicates node type).*
- *Name (name of the node).*
- *Title.*
- *Language (only for documents).*
- *Size.*
- *Update date.*
- *Author.*
- *Version (only for documents).*

*Also, a browser table can show extra columns based on metadata field values. For more information about how to enable it, take a look at: **Administration** > Profiles.*

*When you right-click a node, a **contextual menu** is shown. You will have distinct menu options depending on the selected node type and the visible stack panel combination.*

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

A complete description of all menu options is available at: Working with the menu bar.

*Available menu options when **Taxonomy, My documents** stacks are visible:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*
- *Lock.*
- *Unlock.*
- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Convert.*
- *Add a note.*
- *Add category.*
- *Add keyword.*
- *Add a metadata group.*
- *Update metadata group.*
- *Add a bookmark.*
- *Set the default home.*
- *Export to ZIP.*
- *Send the link.*
- *Send a document attachment.*
- *Forward e-mail.*

*Available menu options when **Categories and Metadata** stacks are visible:*

- *Download.*
- *Edit.*
- *Update.*
- *Cancel edit.*
- *Lock.*
- *Unlock.*
- *Delete.*

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

- *Rename.*
- *Add a note.*
- *Add category.*
- *Add keyword.*
- *Add a metadata group.*
- *Update metadata group.*
- *Go to (document, folder, mail, record). Open the node location in taxonomy view.*
- *Send the link.*
- *Send a 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 a note.*
- *Add category.*
- *Add keyword.*
- *Add a metadata group.*
- *Update metadata group.*
- *Merge PDF.*
- *Export to ZIP.*
- *Create from a template.*
- *Add a bookmark.*
- *Send the link.*
- *Send a document attachment.*
- *Forward e-mail.*

Creating from a template *helps you create new documents based on existing templates. More information: Creating templates.*

Available menu options when the ***Email*** *stack is visible are:*

- *Delete.*
- *Rename.*
- *Move.*
- *Copy.*
- *Convert.*
- *Add a note.*
- *Add category.*
- *Add keyword.*
- *Add a metadata group.*
- *Update metadata group.*
- *Merge PDF.*
- *OCR data capture.*
- *OMR data capture.*
- *Send the link.*
- *Send a document attachment.*
- *Forward e-mail.*

Available menu options when the ***Trash*** *stack is visible are:*

- *Restore.*
- *Purge.*

The ***checkbox*** *helps you to* ***select several nodes*** *at the same time and* ***apply an action to all****.*

You can perform actions on several nodes at the same time *(you have a checkbox in each row for selecting the node). The bulk actions allowed are:*

- *Delete.*
- *Move.*
- *Copy.*
- *Add notes.*
- *Add categories.*
- *Add keywords.*
- *Add metadata groups.*

- *Change metadata group field values.*

When you subscribe to a node, an ***icon*** *will appear at 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 of in the user profile. See Profile Notification tab for more information.*

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 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 mails	*Selects all the nodes of type mail.*
Select all records	*Selects all the nodes of type record.*
Deselect all	*Unselect all the nodes.*

Pagination

When ***the pagination feature*** *is enabled, an extra toolbar appears:*

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 the previous page.*
Go to next	*Goes to the next page.*
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 the grouped properties or features, like a preview associated with the selected node. The selected node can be a navigator node or a browser node.

> *Basically, the selected node in the navigator panel determines the content shown in the browser panel, and depending on whether the last node selected by the user was in the navigator or browser panel, the properties*

panel will show the properties of the active node from the navigator or browser.

The features described below are the default features. Other features not described here may be shown in your properties tab 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 users subscribed to the node***. *For quick node sharing, there are two special icons named* ***URL*** *and* ***WebDAV****; clicking them copies 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 versions of the document.* *From this view you can:* • ***Restore to a previous version*** *of the document.* • ***Preview any document version*** *on the right panel.* • ***Compare two versions of the document*** *on the preview panel.* • ***Compact the history***. *The action of compacting the history will remove all document versions except the active one. This action reduces repository size but must be taken into consideration because it cannot be reversed.*
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, mails, or records). From this view you can manage the relations of the document, adding or removing relations.*

To set a new relation:

- *First, **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 provides 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 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 an audit trail point of view.

	*This option is only available if you have enabled the **Activity log** feature.*
Forum	*Displays a forum associated with the document.* *From this view you can:* • ***Add a new topic***. • ***Post a reply*** *of the topic.* • ***Quote a reply*** *of the topic.* • ***Edit a reply*** *of the topic.* • ***Delete a topic***. *To delete or edit a post in the topic you must be the owner or have administrator privileges.* *This option is only available if you have enabled the **Forum** feature.*

Tabs on nodes of type folder

Tab	*Description*
Properties	*Displays the properties of the selected folder.* *From this view you can:* • ***Change the folder style***. • ***Set the keywords*** *of the folder.* • ***Change the title*** *of the folder.* • ***Set relations between the folder and categories***. *This tab shows a **list with the users subscribed to the node**.* *For a quick node sharing, there are two special icons named **URL** and **WebDAV** when clicking on them, it will copy in your clipboard a browser or WebDAV URL.* *More information: Folder style.*
Notes	*Displays a list of notes related 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 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 an 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 of the topic you must be the owner or have administrator grants.* *This option is only available if you have enabled the **Forum** 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, it will copy in your clipboard a browser or WebDAV URL.*
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, mails or records). From this view you can manage the relations of the document, adding or removing relation.* *To set a new relation:* • *First must **select the type** from the relation type list.* • *Then **choose one** of the defined relations.* • ***Select a node in the repository**, for it are shown four icons for searching a document, folder, mail or record.* *There 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 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 an 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.*

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 is shown as a **list of users subscribed to the node**.* *For quick node sharing, there are two special icons named **URL** and **WebDAV**; clicking them copies 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, mails or records). From this view you can manage the relations of the record, adding or removing relation.

To set a new relation:

- *First must **select the type** from the relation type list.*
- *Then **choose one** of the defined relations.*
- ***Select a node in the repository**, four icons for searching are shown document, folder, mail or record.*

There 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 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 an 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.*

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.*

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

CTRL+G	*Show UI testing pop-up.*
CTRL+Z	*Show UI log.*
CTRL+N	*Create a folder.*
F5	*Refresh 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:

- *Search 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 performed, 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. With a double-click on a stored search, you can execute the search again. Right-clicking on 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 provides access to different searches to query the content of the application:

- ***Default search** is a full-text search. Default search is also available from the top right corner on the screen.*
- ***Advanced search** enables you to search on keywords, metadata and additional criteria such as location in the application.*

*You can **switch between default and advanced** search modes 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 of reducing inflected (or sometimes derived) words to their word stem, base, or root form (generally a written word form). The stem need not 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. Stopwords are removed from searches. For example "the", "is", "at", "which" and "on" are stopwords.*

Operators and wildcards

*By default, the **search engine uses the "AND" operator** internally.*

You can use wildcards:

- *Add an asterisk "*****" as a placeholder for any unknown or wildcard terms, for example, "***2015**", "**2015***", "**20*15**" or "***2015***".*
- *Add 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 "*****" for you at the end.*

Default search

Default search is a full-text search that searches only 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.

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 in 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 type document have a language value.*	*Yes*

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

User	*The user who created the node.*	*No*
Date range	*The range of dates during which the node was modified (or created).* *Each selected date is set to 00 hours, 00 minutes, 00 seconds and 00 milliseconds. That means if you want to query for 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 advanced view :

Field	***Description***	***Wildcard allowed***
Folder	*Restrict results to folders.* *When the Recursive checkbox is checked, subfolders are also considered.*	*No*
Category	*Restricts results to categories. The nodes must be associated to a specific category.*	*No*
Type	*Restricts 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 'Subject' mail field.* *This field is only available if you have **checked the Mail checkbox**.*	*Yes*

In 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 almost all 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 search view is shown.*
Results compact view checkbox	*When checked, the results are shown in a compact table with minimal details.*

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, it will be saved as a "stored search" or "user news".*
Page results lists	*Set the limit of nodes to be shown in the results list.*
Clean button	*Resets the search.*
Search	*Executes the search.*
Results	*Indicates the 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 results.*
Export to ZIP	*Downloads a ZIP file with results.*

Results

The Results panel at the bottom center shows a table of nodes (documents, folders, mails, and records) that 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 on the compact view, a contextual menu is shown with the options:

Feature	*Description*
Download	*Downloads the node.* *Only shown in document or mail nodes.*
Go to document	*Goes to the document location.* *Only shown in document nodes.*
Go to folder	*Goes to 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.*

In full view you see detailed information for each node and some features are also available:

Feature	*Description*
Download	*Downloads the node.* *Only shown in document or mail nodes.*
Notes	*Shows notes. From this view you can manage the notes of the node.*
Preview	*Previews the node.* *Only shown in document or mail nodes.*
Options	*Shows extra 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 the dashboard

The dashboard panel provides a set of dashboards. Each user has a set of personal dashboards that display information relevant to them and help provide a global view of application activity.

The dashboard shows:

- *Displays the actions that have been carried out in the system*
- *Displays a global view of application activity.*
- *Displays information based on the scheduled user search queries.*

The results listed in the dashboard boxes depend on the security access level for 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 subscriptions to documents by the user.*
Subscribed folders	*Lists subscriptions to folders by the user.*
Subscribed mails	*Lists subscriptions to emails by the user.*
Electronic mails	*Displays a list of the last 20 emails uploaded by the user.*
Attachments	*Displays a list of the last 20 documents attached to emails uploaded by the user.*
Last modified documents	*Displays a list of the last 20 documents modified by the user.*

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

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 review 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.

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 was 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.

Digital Signature viewer extension

*Digital Signature viewer helps you to **view** and **verify** signed documents.*

*This option is only available if you have enabled the **Digital Signature viewer** extension.*

More information at Enable extensions.

This extension requires additional software configuration.

Verify a signed document

- *Choose a signed document.*
- ***Show the Digital Signature** viewer tab.*

Features:

Feature	*Description*
Integrity	*Display the result of checking the digital signature.*
Trust	*Display the result of checking whether the signer's certificate is trusted.*
Revocation	*Display the result of checking the certificate's revocation list.*
Validity	*Display the result of checking the certificate validation or the signing time.*
Download the signed version	*You can download the signed document.*
Certificate information	• *You can download the certificate.* • *Display the subject.* • *Display the issuer.*
Validity information	*Display the signing time.*
Signed properties	• *Display the signing reason.*

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

	• *Display the signing location.*

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.*

There are several parameters passed in the URL request:

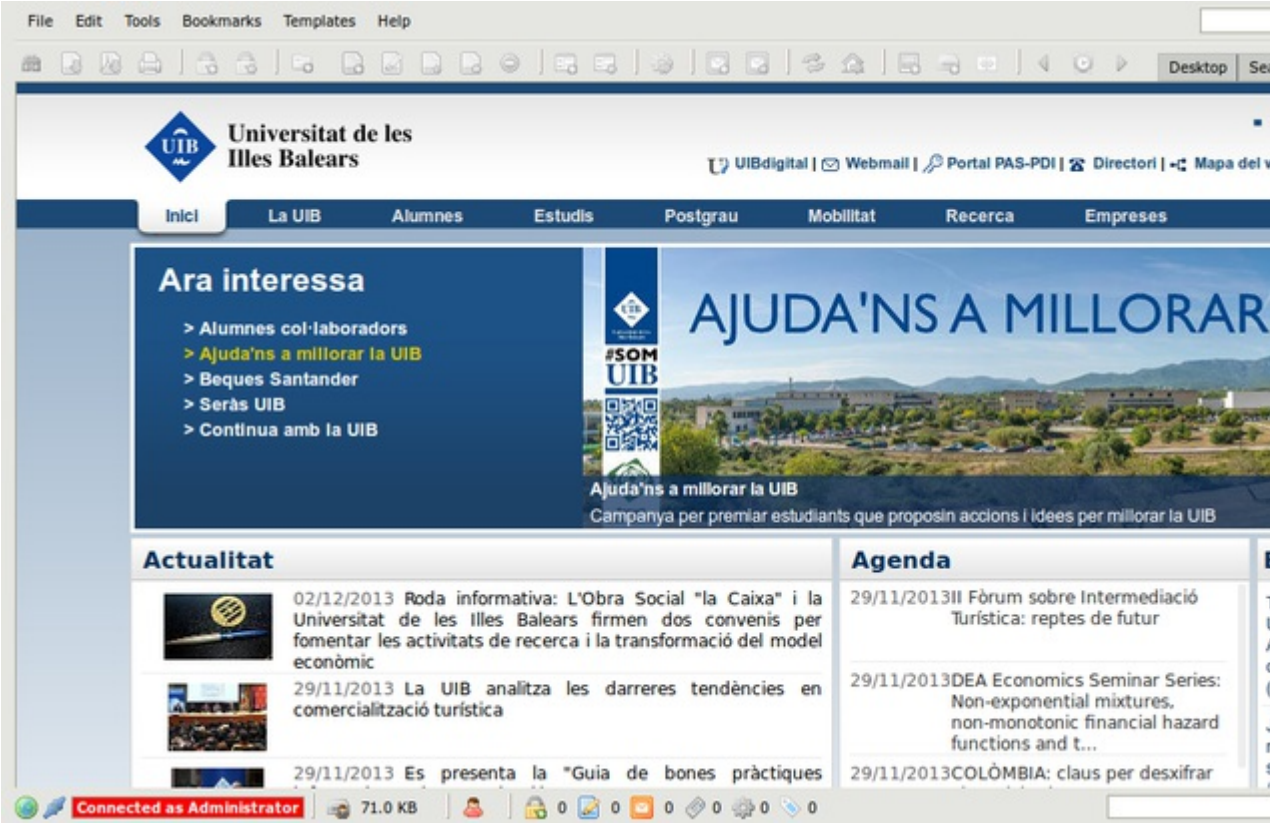
- *The authentication **token** (encrypted)*
- *The **lang** (language ID)*
- *The node **UUID***
- *The **user ID***

Configuration

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 http**, it's considered an **OpenKM relative path** and will be merged with your OpenKM server URL like **YOUR_OPENKM_URL/url***
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.extra.tab.workspace**" accessible to all users, the value "**extension.extra.tab.workspace**" must be added to the current list.* *If you forget to add the property name, only administrators will be able to get the value, so some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

Sample


File Edit Tools Bookmarks Templates Help
Desktop
Universitat de les Illes Balears
UIBdigital | Webmail | Portal PAS-PDI | Directori | Mapa del
Inici La UIB Alumnes Estudis Postgrau Mobilitat Recerca Empreses
Ara interessa
> Alumnes col·laboradors
> Ajuda'ns a millorar la UIB
> Beques Santander
> Seràs UIB
> Continua amb la UIB
AJUDA'NS A MILLORAR
#SOM UIB
Ajuda'ns a millorar la UIB
Campanya per premiar estudiants que proposin accions i idees per millorar la UIB
Actualitat
02/12/2013 Roda informativa: L'Obra Social "la Caixa" i la Universitat de les Illes Balears firmen dos convenis per fomentar les activitats de recerca i la transformació del model econòmic
29/11/2013 La UIB analitza les darreres tendències en comercialització turística
29/11/2013 Es presenta la "Guia de bones pràctiques
Agenda
29/11/2013 II Fòrum sobre Intermediació Turística: reptes de futur
29/11/2013 DEA Economics Seminar Series: Non-exponential mixtures, non-monotonic financial hazard functions and t...
29/11/2013 COLÒMBIA: claus per desxifrar
Connected as Administrator
71.0 KB

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 as request parameters 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 consists of a pair of values declaring tab content; the format is:* ***name;url*** *name1;http://localhost:8180/openkm/test.jsp* *name2;http://www.openkm.com*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.external.tab.document**" accessible to all users, the value "**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

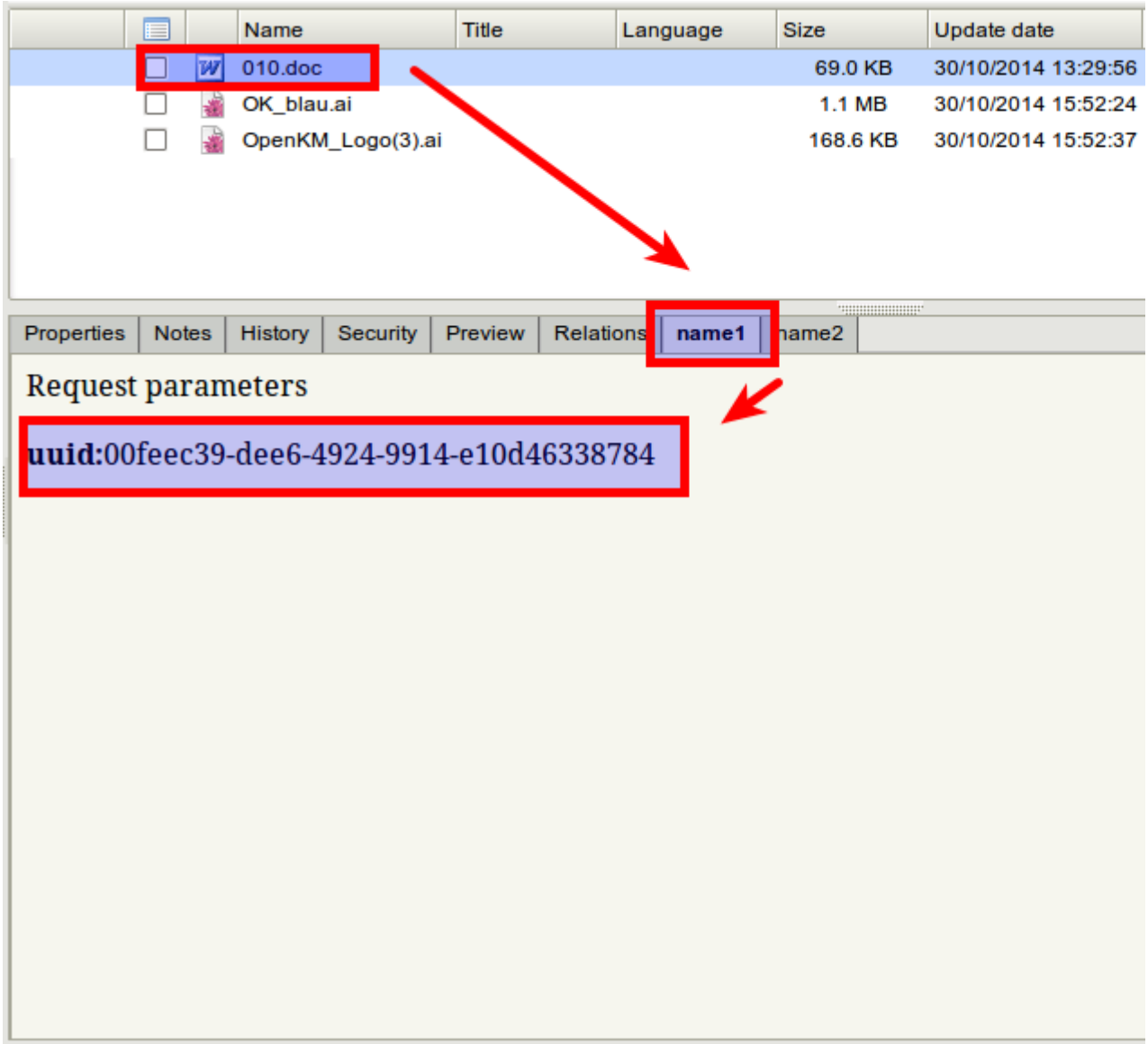
Configuration

Create a file named test.jsp and copy it to the $TOMCAT_HOME/webapps/openkm folder.

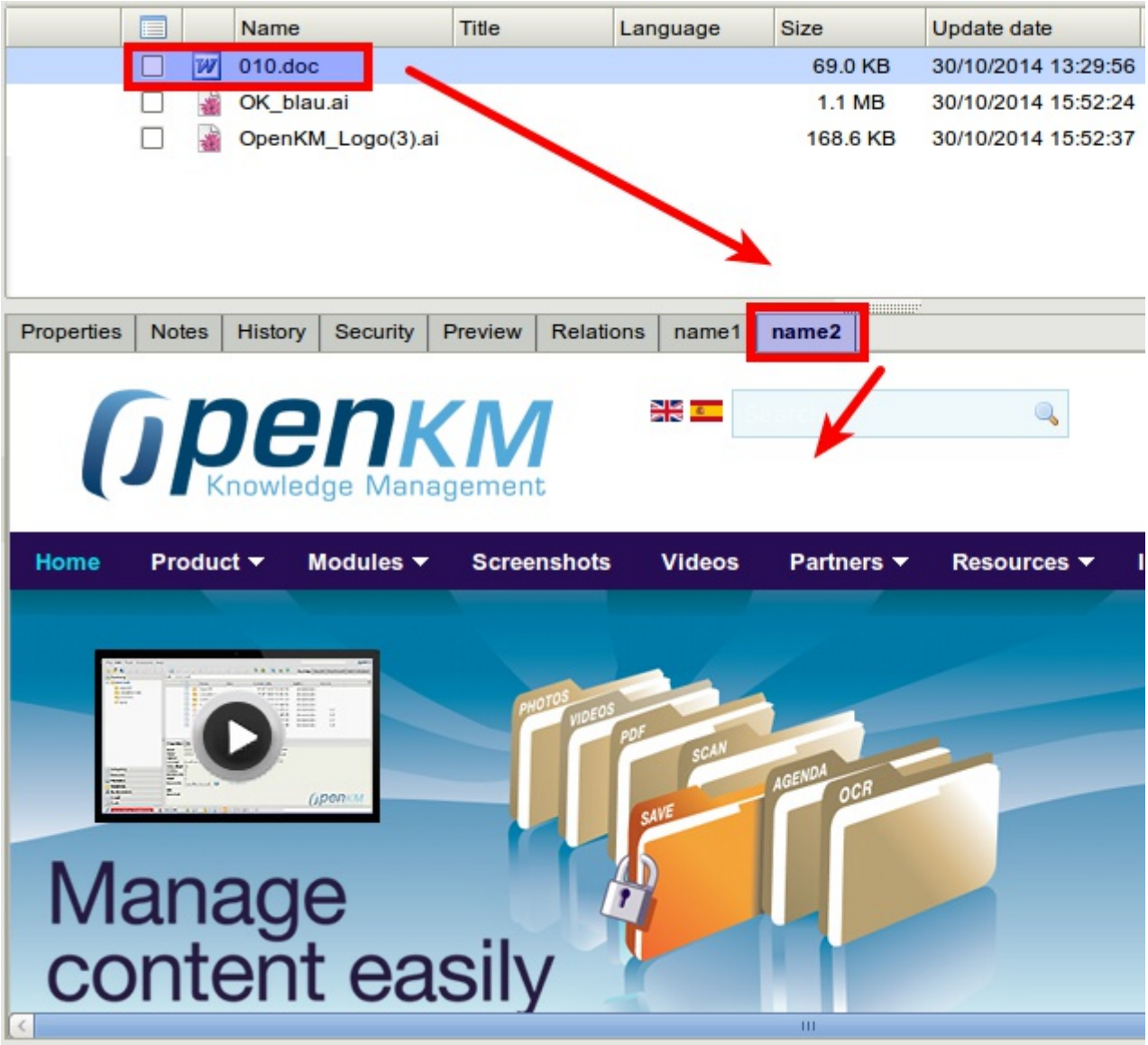
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External 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*** *the "**name1**"* ***tab***.



- ***Click on** the "**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 node folder 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 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`

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, so some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

Sample

Configuration

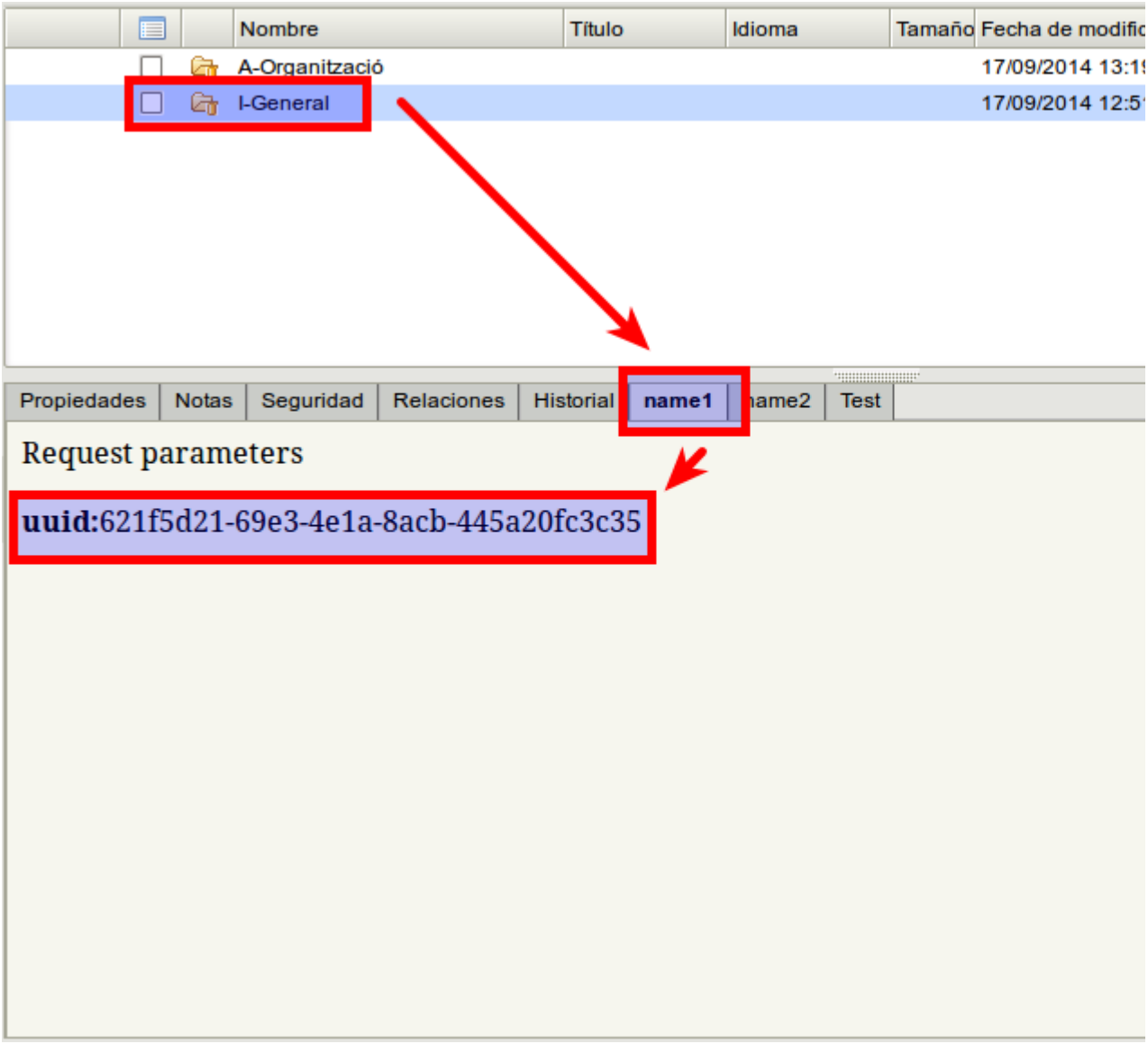
Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

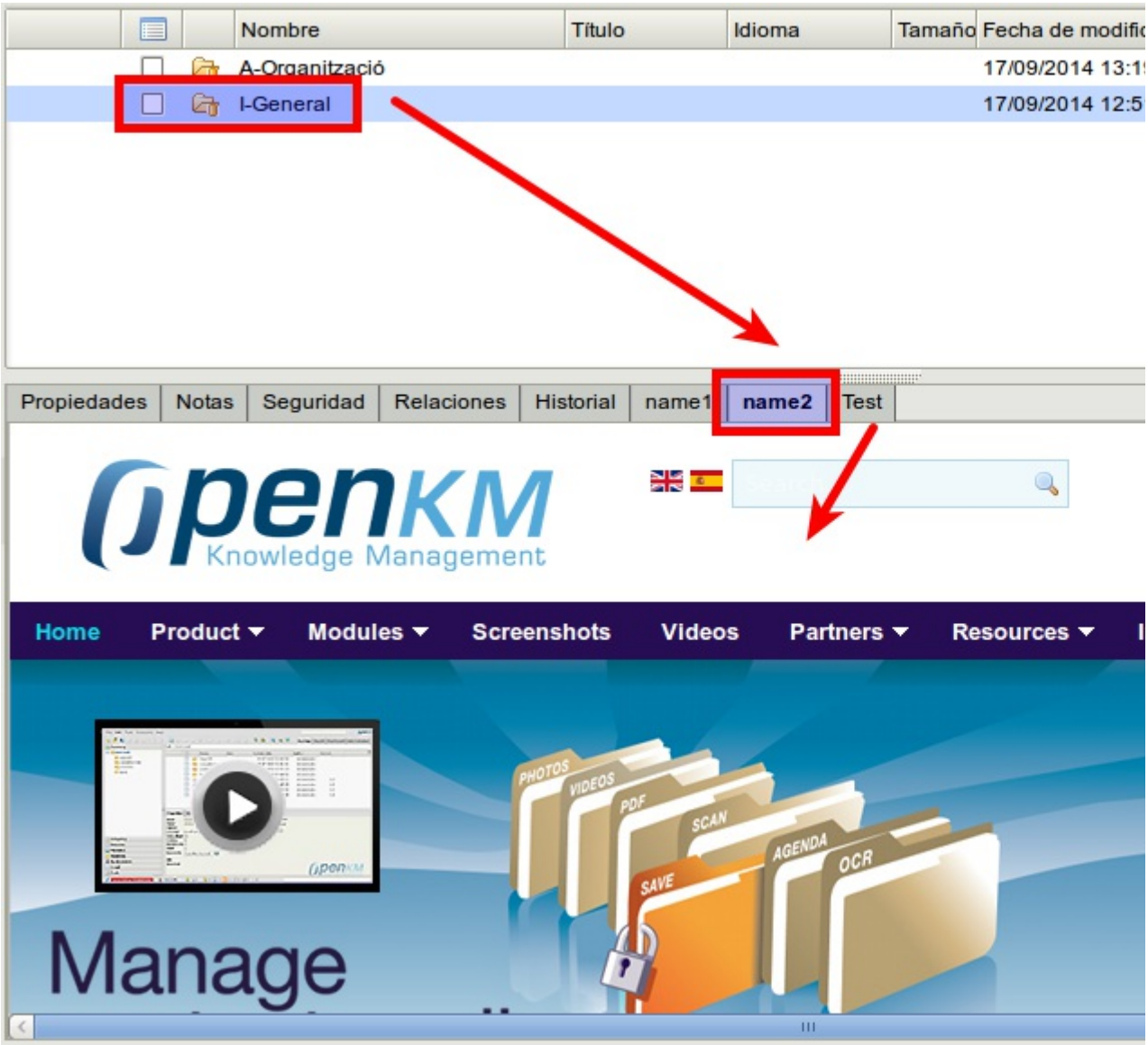
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a folder***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab.***



External tab mail extension

The External Tab Mail feature helps you show the tab content in the properties tab based on an external URL request when a node mail is selected.

*This option is only available if you have enabled the **External tab mail** feature.*

There are several parameters passed in the URL request:

- *The authentication **token** (encrypted)*
- *The mail **uuid***
- *The **userId***

Configuration

Field / Property	*Type*	*Description*
extension.external.tab.mail	***List***	*Each row contains 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*
webservices.visible.properties	***List***	*To make the values of the parameter "**extension.external.tab.mail**" accessible to all users, the value "**extension.external.tab.mail**" 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

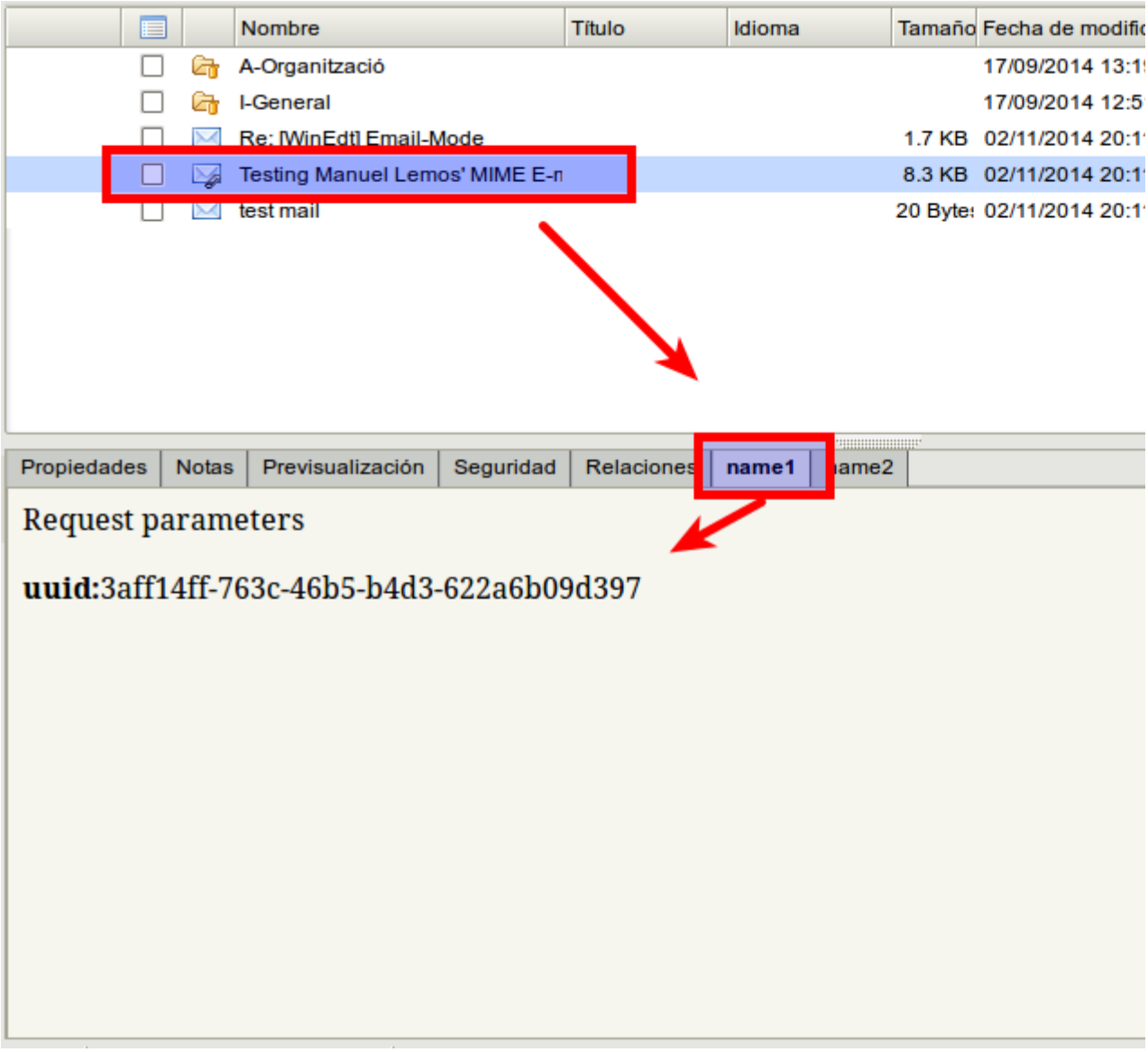
Configuration

Create a file named test.jsp and copy it to the $TOMCAT_HOME/webapps/openkm folder.

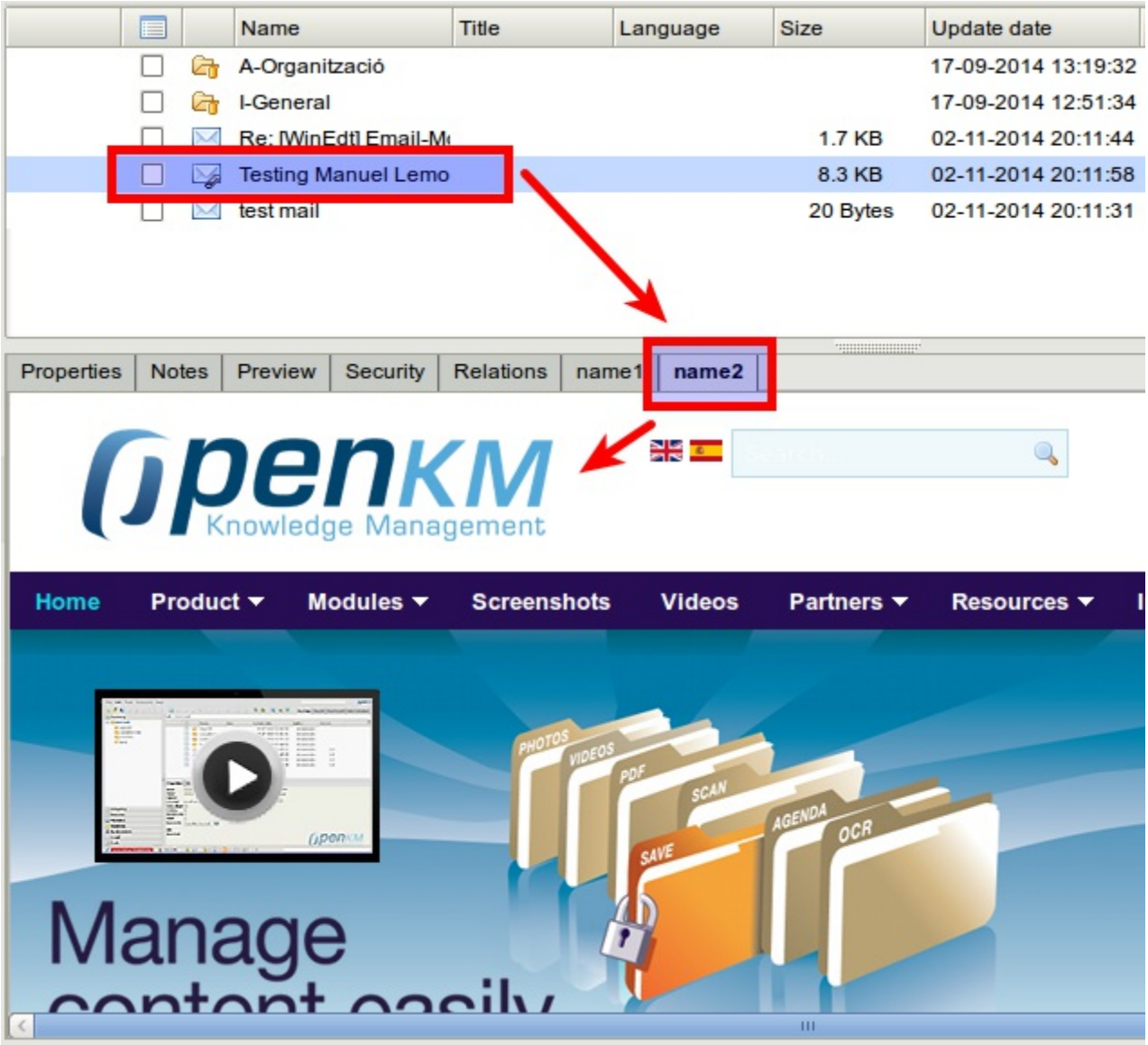
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a mail***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab.***


Name
Title
Language
Size
Update date
A-Organització
17-09-2014 13:19:32
I-General
17-09-2014 12:51:34
Re: [WinEdt] Email-M
1.7 KB
02-11-2014 20:11:44
Testing Manuel Lemo
8.3 KB
02-11-2014 20:11:58
test mail
20 Bytes
02-11-2014 20:11:31
Properties
Notes
Preview
Security
Relations
name1
name2
OpenKM
Knowledge Management
Search
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 the 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 contains 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*
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 get the value, so some extensions may produce errors for regular users.*

More information is available at Configuration parameters.

Sample

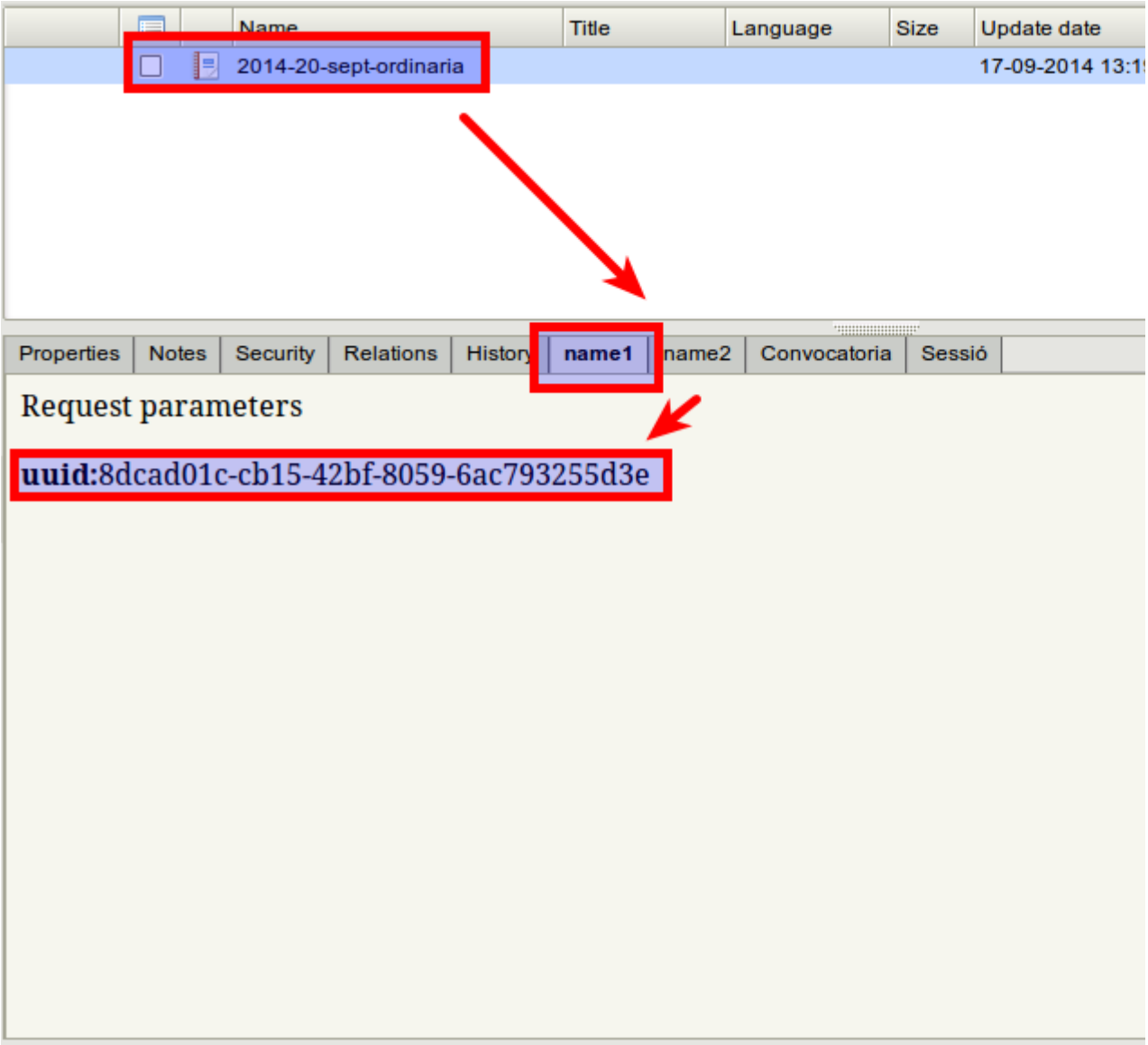
Configuration

Create a file named test.jsp and copy it into the $TOMCAT_HOME/webapps/openkm folder.

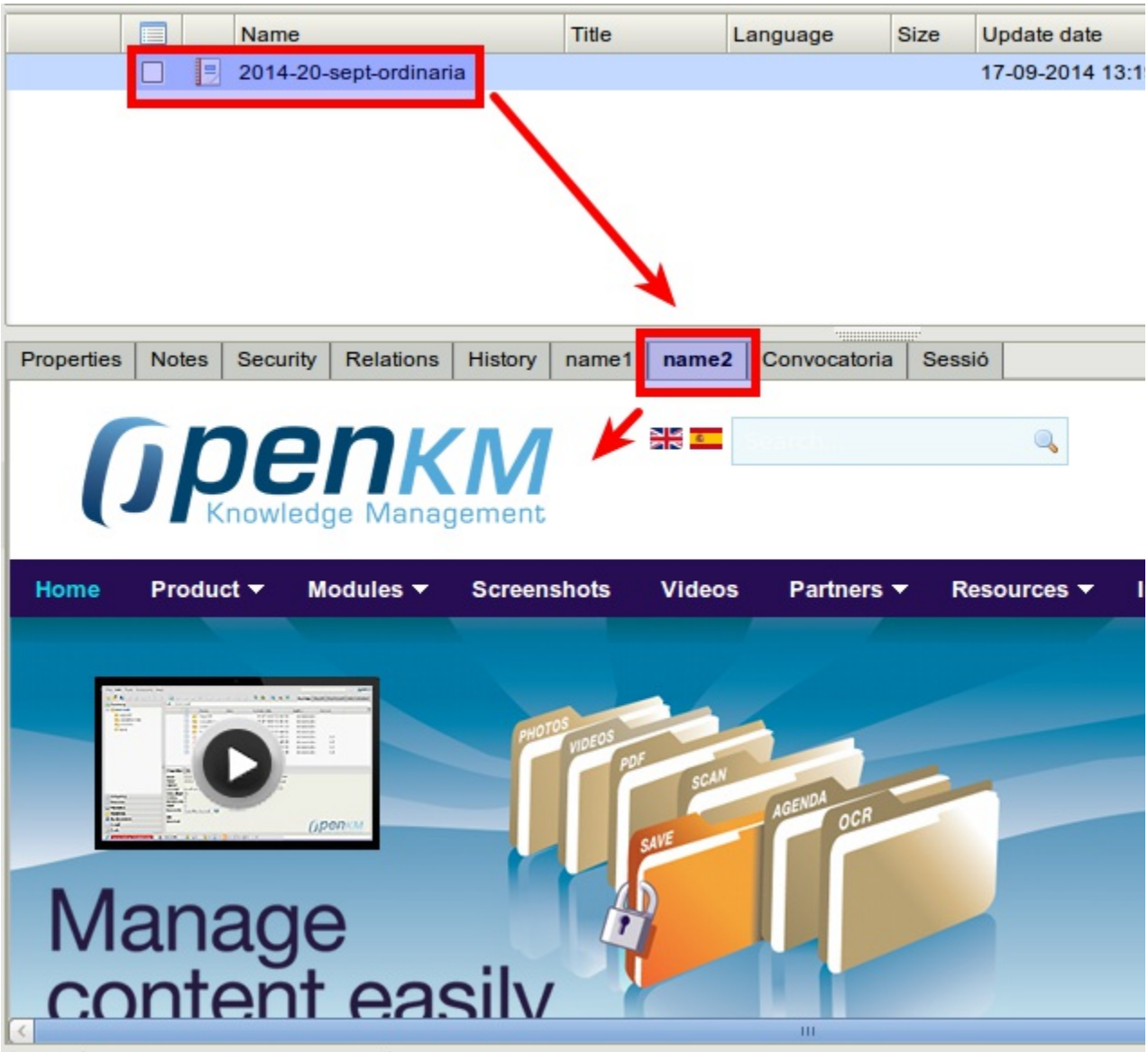
```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-tr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>External URL request example</title>
</head>
<body>
 Request parameters
 <br />
 <br />
 <b>uuid:</b><%=request.getParameter("uuid")%><br />
</body>
</html>
```

Usage

- ***Choose a record***.
- ***Click on "name1" tab***.



- ***Click on "name2" tab***.


Name
Title
Language
Size
Update date
2014-20-sept-ordinaria
17-09-2014 13:1
Properties
Notes
Security
Relations
History
name1
name2
Convocatoria
Sessió
OpenKM
Knowledge Management
Home
Product
Modules
Screenshots
Videos
Partners
Resources
PHOTOS
VIDEOS
PDF
SCAN
AGENDA
OCR
SAVE
Manage
content easily

Forum extension

The Forum extension helps you create discussion topics on nodes (documents, folders, mail, or records). Also, there is a general-purpose forum available from the Dashboard view.

*This option is only available if you have enabled the **Forum** feature.*

Only administrators can create, update, or delete forums from the Dashboard view.

Only administrators can view the "document discussions" forum in the Dashboard view. This forum contains all the node topics.

No one can delete the "document discussions" forum in the Dashboard view.

Create a new topic

- ***Choose a node** (document, folder, mail, or record).*
- *Click on the **New topic button**.*
- ***Fill in the form**.*
- *Click on the **Create button**.*

Create a reply

- ***Choose a topic on a node**.*
- *Click on the **Post reply button**.*
- ***Fill in the form**.*
- *Click on the **Create button**.*

Update a post

- ***Choose a topic on a node**.*
- *Click on the* ***Edit icon**.*
- ***Modify the form**.*
- *Click on the **Update button**.*

Delete a 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.

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

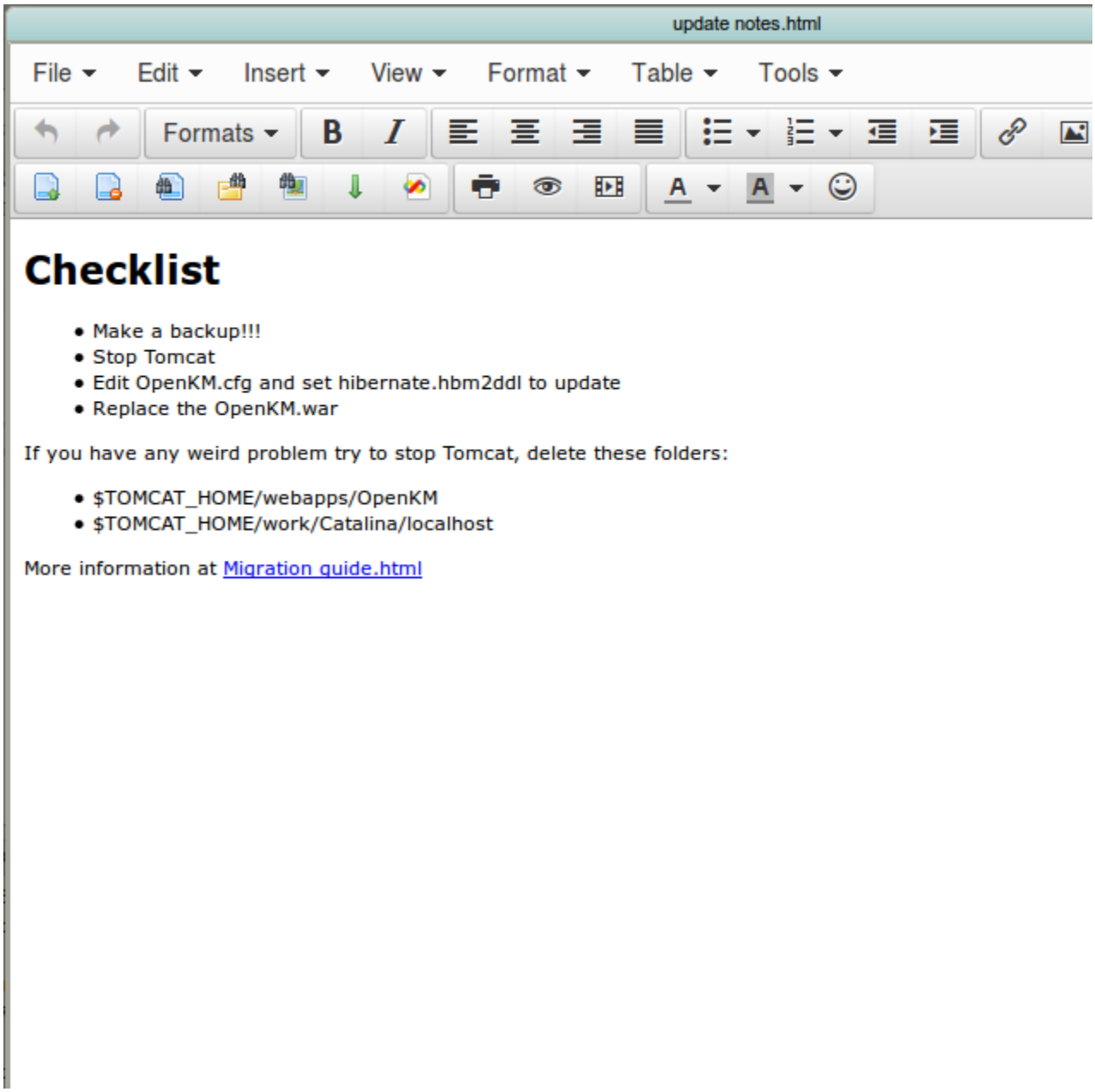
Feature	*Description*
Update	*Upload a new document version (check-in).* *When uploading a new version of a document, you can notify users about document changes. The application sends the notification by mail.*
Cancel edit	*Cancel document editing.*
Link document	*Add a link to a document.*
Link folder	*Add a link to a folder.*
Add image	*Add an image.*
Download link	*Add a download link.*
Code	*Add a code section.*

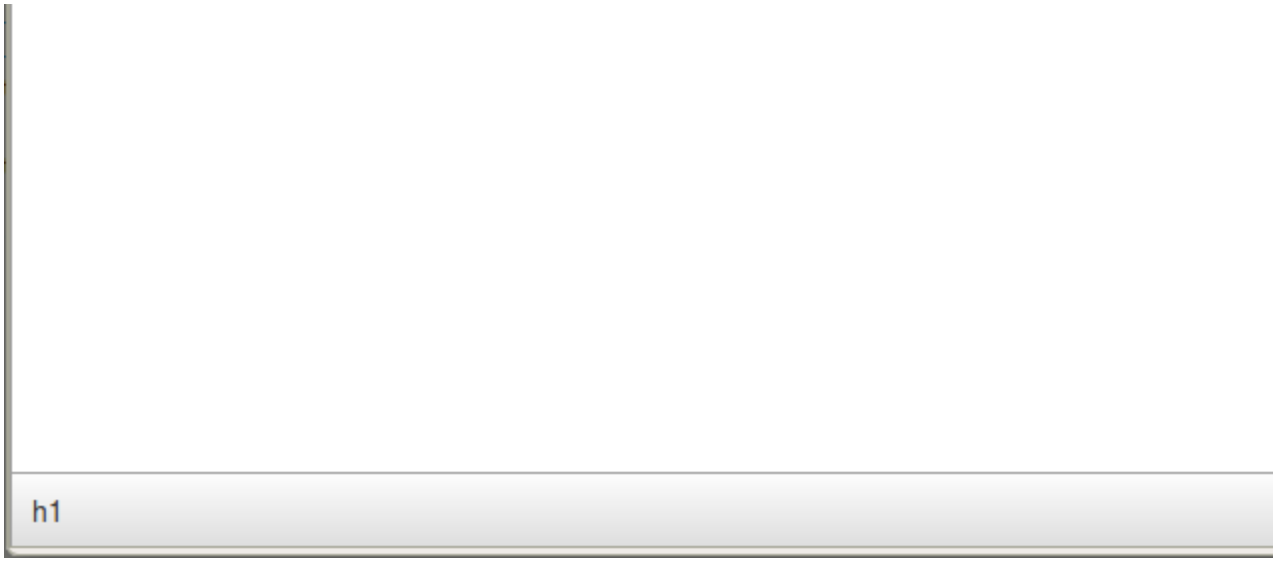
You can navigate through linked documents and folders or download documents from the Preview view.

Edit HTML

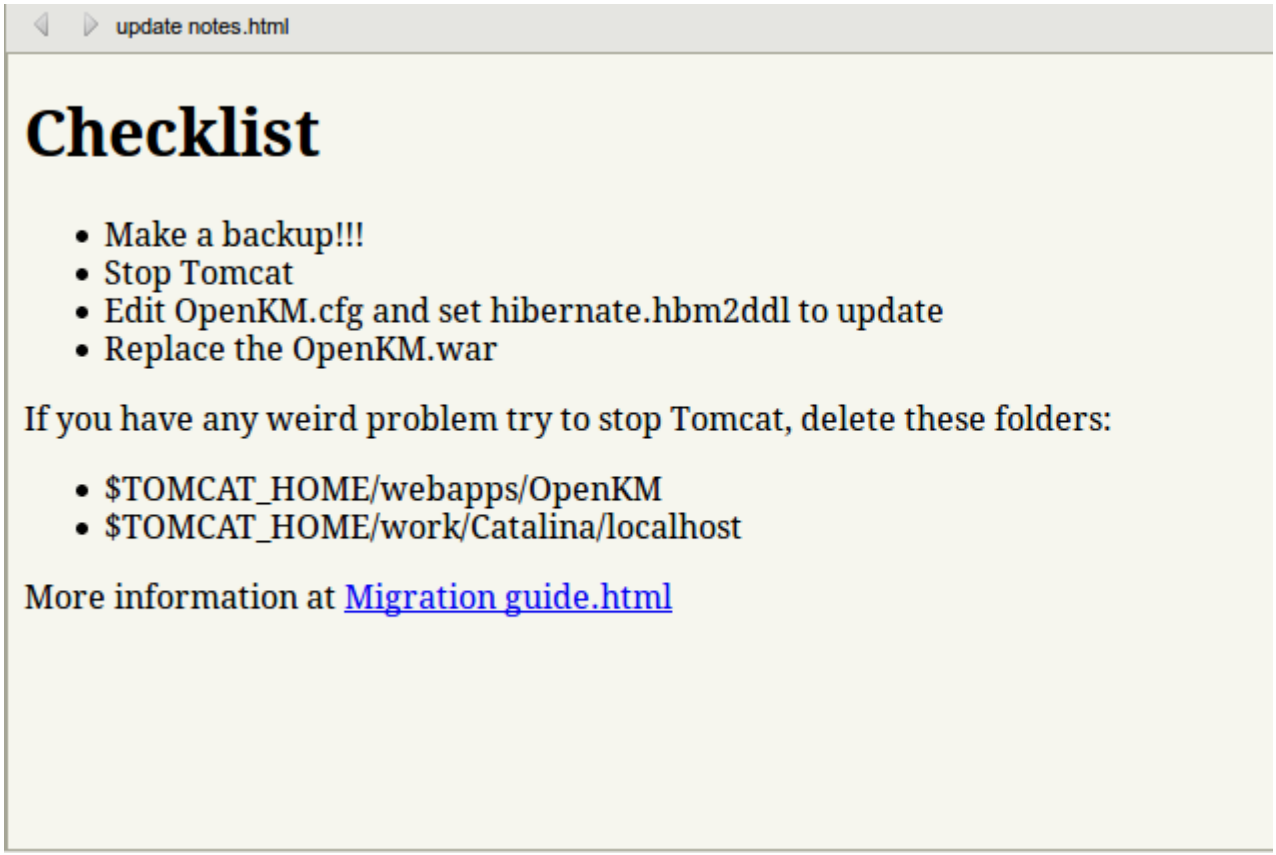
- ***Choose an HTML document***.
- *Click the* ***Edit icon***.
- *Modify the content.*
- *Click the* ***Update icon***.

Editor panel





Preview panel



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

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 Office Add-in.*

Scanner Client

Scanner Client allows you to capture images from scanners and upload them to the OpenKM repository.

> *Optionally:*
>
> - *The images can be converted to a PDF file.*
> - *You can add a metadata group while uploading files to the OpenKM repository.*
>
> *More information at Scanner Client.*

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

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 OpenKM Microsoft Outlook Add-in you can save emails to the OpenKM repository from within Microsoft Outlook.

*Importing emails creates a folder structure based on the received email date (**year/month/day**) and imports them by default to "**/okm:mail/{userId}**" node.*

*When a user edits a document, it is saved locally in "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM**".*

When editing OpenKM documents, a user can decide to upload or cancel editing. After either operation, the local file is removed so as not to cause user confusion. When you upload a new file, it is not removed from your computer.

*In the file "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/openkm.xml**", the OpenKM connection data are saved. The password is encrypted.*

*In the file "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/document.xml**", the document editing information and status are saved.*

There is a log file for each application at:

- *"**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/WordAddin.log**".*
- *"**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/ExcelAddin.log**".*
- *"**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/PowerPointAddin.log**".*
- *"**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/OutlookAddin.log**".*

*Microsoft Office Add-in is **compatible with versions 2000 to 2013**.*

Configuring the add-in

- *Click on the configuration toolbar icon (menu option in Microsoft Outlook).*
- *Set version to "**Webservices 2.0**".*
- *Choose a language (by default English).*
- *Set the **username**.*
- *Set the **password**.*
- *Set the **host** (for example, http://localhost:8080/OpenKM).*
- *Click the **Test** button to check the connection configuration.*

Editing documents from Microsoft Office

*Depending on your Microsoft Office version, you will see an OpenKM **toolbar** or a **Ribbon**.*

Feature	*Description*
OpenKM ***Configuration***	*Configure the connection to the OpenKM server.*
Add document	*Upload a new document to the OpenKM repository.*
Edit	*Edit document (check-out).* *When a user edits a document, it is saved locally in "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM**".* *You can continue editing the document another day (the document status is saved in the "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/document.xml**" file).*
Update	*Upload a new document version (check-in).*
Cancel edit	*Cancel document editing.*
Help	*Open the OpenKM Microsoft Office Add-in URL in your default web browser.*

Importing emails from Microsoft Outlook

- ***Choose** one or more **emails**.*
- ***Click on the import** menu option.*
- ***Follow the wizard** to select the destination folder. When the grouped checkbox is enabled, the **year/month/day** folder structure will be created.*
- *Click the **Upload** button.*

Add new translation

*The translation file is located at "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/OKMAddin-enGB.properties**". The easiest way is to change the contents of the file directly.*

- *Create a new translation file at "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/**" (for example, copy OKMAddin-enGB.properties to OKMAddin-esES.properties for the Spanish translation).*
- *Set the translations.*
- *Edit "**%USER_LOCAL_DOCUMENT_FOLDER%/OpenKM/openkm.xml**" and change the tag*

```
<idioma>enGB</idioma>
```

to

```
<idioma>esES</idioma>
```

Known issues

In Microsoft Office 2010 we have detected some issues. If you experience problems using this version, try the following:

- *Open the menu **File** > **Options**.*
- *Select **Trust Center** and click on "**Trust Center Settings**" button.*
- *Open **Protected View** option and **disable the first three** checkboxes.*

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

Scanner client

Scanner Client allows you to capture images from scanners and upload them to the OpenKM repository.

Optionally:

- *The images can be converted to PDF files.*
- *You can add a metadata group while uploading files to the OpenKM repository.*

	Feature	Description
	Scan	*Starts the scanning process.*
	Convert to PDF	*Convert an image or a group of selected images to a PDF document.*
	Refresh	*Refreshes the tree.*
	Upload	*Uploads a document.*
	Delete PDF page.	*Deletes a page from the selected PDF document.*
	Add page to PDF	*Adds a new page to the selected PDF document.*

Requirements

Before running the application, make sure you have installed:

- *Visual C++ Redistributable for Visual Studio*
  - *.NET 4.0: Visual C++ Redistributable for Visual Studio 2012 (x86 or x64) (Recommended).*
  - *.NET 2.0: Visual C++ Redistributable for Visual Studio 2008 (x86 or x64).*
- *Adobe Acrobat Reader.*
- *Microsoft .NET Framework 3.5 or higher.*

Configuring

- *Click on **File** > **Configuration**.*

- *Set version to "**Web Services 2.0**".*
- *Choose language (by default, English).*
- *Set the **username**.*
- *Set the **password**.*
- *Set the **host** (for example http://localhost:8080/OpenKM).*
- *Click on the **Test** button to check the connection configuration.*

Compression CCITT Group4

- *Click on the **Configuration** > **Compression group4** menu option.*
- *Check the **black and white CCITT Group4 compression** checkbox.*
- *Set the **percentage** (Threshold); by default it is set to 80%.*
- *Click on the **Accept** button.*

Setting destination folder

- *Click the top-right **Change** button.*
- *Select the destination folder and click the **Select** button.*

Setting scanner configuration

- *Click on Configuration / **Scanner configuration**.*
- *Change the configuration parameters.*
- *Click on the **Accept** button.*

We suggest TWAIN protocol rather than WIA for better scanner feature support.

Scanning

- *Click on the **Scanning** button.*
- *Follow the wizard.*

*A folder based on the date (**year-month-day**) will be created on your server at **My images/OKMScan/Images**.*

Convert scanned images to PDF

- ***Choose one or several image files** from the tree.*
- *Click on the **Images to PDF** button.*

*A PDF file will be created in a folder based on the date (**year-month-day**) on your server at **My images/OKMScan/Pdfs**.*

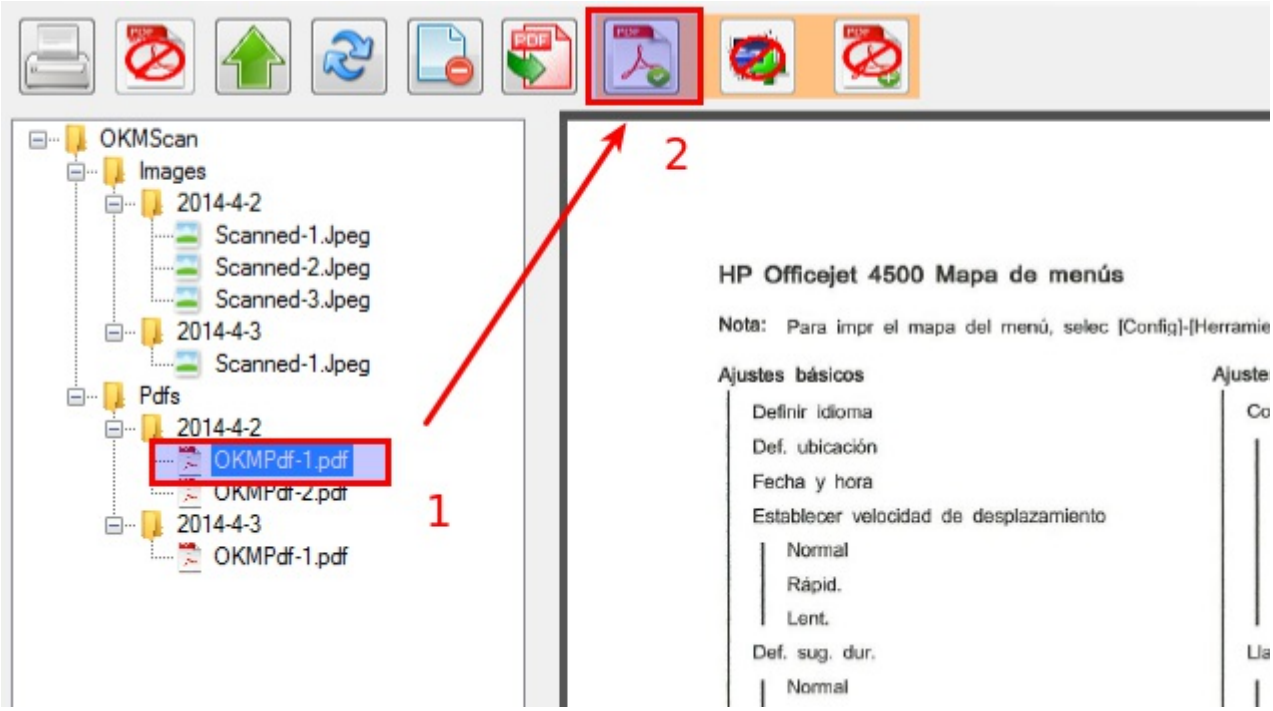
Uploading

- ***Choose one or several files*** *(images or PDFs).*
- *Click on the **Upload** button.*
- ***Optionally, you can set metadata groups**.*
    - ***Choose the metadata group**.*
    - *Click on the **Add** button.*
    - ***Fill the metadata group fields**.*
    - *Click on the **Add** button.*

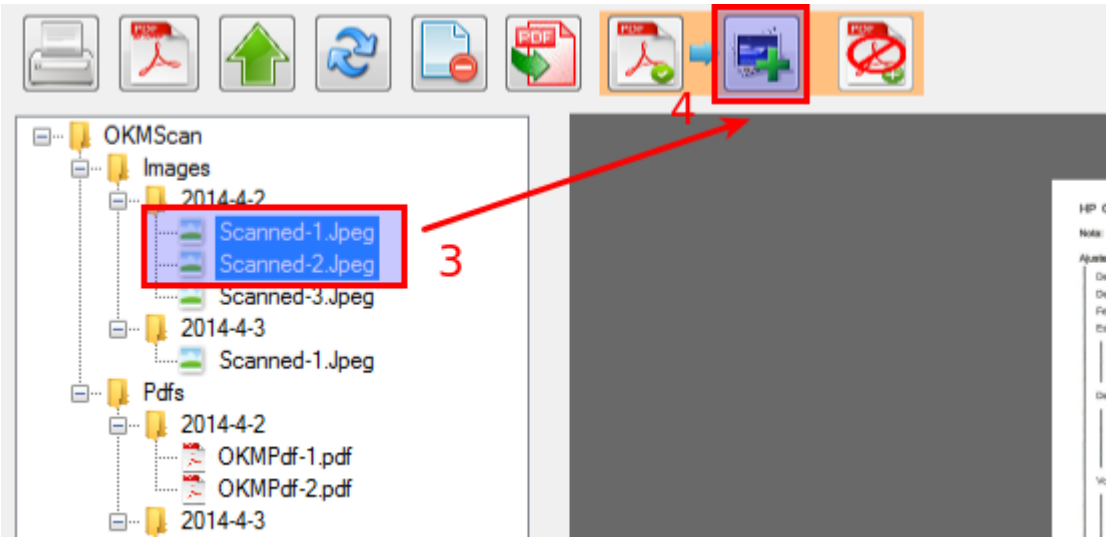
Add images to an existing PDF file

- *Click on the **Add page to PDF** button. Extra icons will be shown in the toolbar.*

- *The action has three steps: first, **select a PDF file** and then click on the icon to **confirm PDF selection**.*



- *The second step is to **select one or several images** and click on the icon to **confirm image selection**.*



- *Finally, click on the icon* *to* ***add the images****. A popup will be shown to confirm the action; click the* ***Yes*** *button.*

Remove pages from existing PDF file

- ***Choose a PDF file****.*
- *Click on the* ***Delete PDF page*** *button.*
- ***Select a range*** *of pages.*
- *Click on the* ***Accept*** *button.*

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

Update guide

A major OpenKM version implies changes to the repository structure. This means that a migration process should be run to apply these changes. Sometimes the database structure is also modified and needs to be updated.

Here you can find the migration processes between different OpenKM releases:

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

Changelog

OpenKM 7.0 Community

v7.0.0

- *Release: NOT RELEASED*

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:

- *8.5.57 and later*

By default, OpenKM is installed with a specific Tomcat version. But every few weeks, a new version is released, and you may want your installation 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 contents of the TOMCAT_HOME/lib folder and replace them with the downloaded content.*
- *Start Tomcat / OpenKM again.*

Once the process has finished, you can verify the new version:

```
$ cd TOMCAT_HOME
$ ./bin/catalina.sh version
```

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 to the destination. Sometimes, when you move data, the applied security might change, which causes issues in the OpenKM startup process.*
>
> *You can set them recursively:*
>
> ```
> $ sudo chown openkm:openkm tomcat-8.5.47 -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 a special installation.*
tomcat-XXX/logback.xml	File	OpenKM log configuration file. It 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** in **/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 will change. In case of Linux, you can check it by doing (as root):

```
$ crontab -l
```

This way, you can see the location of the backup script.

Preserving the old Tomcat is a good idea, but keep in mind that the backup script usually backs up the /home/openkm directory, so move it to another location.

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 the services, check the user executing OpenKM. Usually the user should be "Local Services", but sometimes a standard user is used. Remember to choose the right user when registering the OpenKM service again.*
- *Check 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 with an Administrator user.*
- *Go to tomcat-XXX/bin folder.*
- *Execute the next command:*

```
$:> service.bat remove
```

Second option:

- *Open a terminal with 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 to the destination. Sometimes, when you move data, the applied security might change, which causes 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-	*File*	*OpenKM security configuration file.*

XXX\openkm.xml		*This file might not exist. You can find this file only in a special installation.*
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 a service***

- *Open a terminal with an Administrator user.*
- *Go to tomcat-XXX/bin folder.*
- *Execute the next command:*

```
$:> service.bat install
```

- *Execute the next command:*

```
$:> openkm //US//OpenKM ++JvmOptions="-XX:+UseG1GC"
$:> openkm //US//OpenKM ++JvmOptions="-XX:+UseStringDeduplication"
$:> openkm //US//OpenKM ++JvmOptions="-XX:+HeapDumpOnOutOfMemoryError"
```

```
$:> openkm //US//OpenKM ++JvmOptions="-XX:HeapDumpPath=$CATALINA_HOME"
$:> openkm //US//OpenKM ++JvmOptions="-Djava.net.preferIPv4Stack=true"
$:> openkm //US//OpenKM ++JvmOptions="-Dmail.mime.ignoreunknownencoding=true"
$:> openkm //US//OpenKM ++JvmOptions="-Dfile.encoding=utf-8"
$:> openkm //US//OpenKM ++JvmOptions="-Dsun.jnu.encoding=UTF-8"
$:> openkm //US//OpenKM ++JvmOptions="-Djava.awt.headless=true"
$:> openkm //US//OpenKM ++JvmOptions="-
Dsun.java2d.cmm=sun.java2d.cmm.kcms.KcmsServiceProvider"
```

- ***Start the 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.*

Changes from version 6.3

Changes

- *A lot of improvements and changes in the **REST API**.*
- ***Language detection** is **disabled by default**.*
- *Parameters in **OpenKM.cfg** with value "**on**" have the value "**true**" in **openkm.properties**.*
- *Because we consider the Database storage feature to be conceptually malicious and always providing poor performance, it has been definitively removed. That's why the configuration parameter "**repository.datastore.backend**" has been removed.*
- *The password of the user stored in the database using **MD5** encryption has been changed to **bcrypt** encryption.*
- *The **Security access manager** works with a plugin architecture. More information at Creating your own security access manager and Security configuration.*
- *Deprecated parameter named "**hibernate.search.index.home**". Take a look at Change repository home.*
- *Deprecated configuration parameter named "**ask.drag.and.drop.updates**".*
- *New templates location based on node types:*
  - */okm:templates/documents*
  - */okm:templates/folders*
  - */okm:templates/mails*
- *Templates may be filtered by categories. You must create categories at /okm:categories/templates.*
- *In previous versions, the metadata group was first added empty, and then the metadata values were set. Starting with this version, the metadata group is added with values.*

Deprecated features from version 6.3

- ***The jBPM workflow engine** was removed because it was a deprecated engine.*
- ***SOAP** services have been removed.*
- ***Database metadata value** support has been removed (no longer supports tables **OKM_DB_METADATA_TYPE**, **OKM_DB_METADATA_VALUE** and **OKM_DB_METADATA_SEQUENCE)**.*
- ***OMR** has been removed from the OpenKM core; now it is an external module.*
- ***Joomla integration** has been removed.*
- ***WordPress integration** has been removed.*
- ***Terminal Shell plugin** has been removed.*
- ***Google Drive integration** has been removed.*
- ***Macros extension** has been removed.*
- ***Stapling extension** has been removed in favour of the relations feature.*

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

- ***Contact*** *extension has been removed.*
- ***Wiki*** *extension has been removed.*
- ***Bulk uploader*** *based on Java Web Start has been removed.*
- ***The quota*** *feature has been removed (currently will be implemented as a customization).*

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 you to use our OpenKM portable development environment rather than configuring your own Eclipse, especially for Java beginners.*
>
> *We've tried to provide a development platform that can be set up in a few steps.*
>
> *The OpenKM development environment is provided as a virtual machine. You can download it from https://www.openkm.com/en/download.html. In that section of the website you will find a video that explains how to install it.*

Tools:

- *Eclipse*
- *OpenJDK 8*
- *MariaDB*
- *DBeaver*
- *iReport*
- *Tesseract*
- *LibreOffice*
- *Geany*
- *Firefox*

Code:

- *Automation samples.*
- *Crontab samples.*
- *Form samples.*
- *OCR samples.*
- *Report samples.*

Suggested minimum hardware configuration:

- *2 CPUs*
- *4GB RAM*
- *64-bit architecture*

Javadoc

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

Each OpenKM version has a complete Javadoc available at http://docs.openkm.com/apidoc/.

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

Personal development environment

OpenKM personal development environment is distributed as a virtual machine image (OVA).

The personal development environment comes with everything you need to start working quickly:

- *OS: Linux Mint distribution.*
- *OpenKM application installed.*
- *MariaDB database.*
- *Eclipse*
    - *Code samples*
- *IReport*
    - *SQL samples*
    - *Scripting samples*

Requirements

The virtual machine uses 2 cores and 4 GB of RAM; the host machine should have at least 4 cores and 8 GB of RAM.

> *We suggest stopping most applications while running the virtual machine.*
>
> *We suggest installing on a desktop or a laptop computer, although you can use a laptop with good hardware.*
>
> *We suggest using an SSD to get extra performance in the virtual machine.*
>
> *For some users, we have observed that running the virtual machine on a secondary disk provides extra performance; it seems some motherboards do not like the OS and virtualization on the same hard disk.*

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 button "Download OpenKM Community OVA".*
- *Enter your email address.*
- *Click the "Request" button.*

> *The download link is sent by email.*
>
> *The link can be used only once.*
>
> *You have 24 hours from when the email 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 virtualization software, such as VirtualBox, VMware, or Hyper-V.

> *Because OVA (Open Virtual Appliance) is not 100% compatible across virtualization software, VMware users usually need to make a few modifications to the file or extract the VMDK to import the virtual machine.*
>
> *If you are looking for a quick deployment, we recommend using VirtualBox.*

If you decide to use VirtualBox, follow the steps described in the following video https://www.youtube.com/watch?v=9ugBYf5ggu8&feature=emb_logo

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.*

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 development Java tools like NetBeans.

Requirements

Software:

- *Eclipse*
- *OpenJDK 1.8*

Your Eclipse installation must have support for:

- *Maven*
- *Subversion (recommended)*

Hardware:

Suggested minimum hardware configuration:

- *2 CPUs*
- *4 GB of RAM*
- *64-bit architecture (although you can run it on 32-bit, we do not recommend that 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 (32- or 64-bit).*

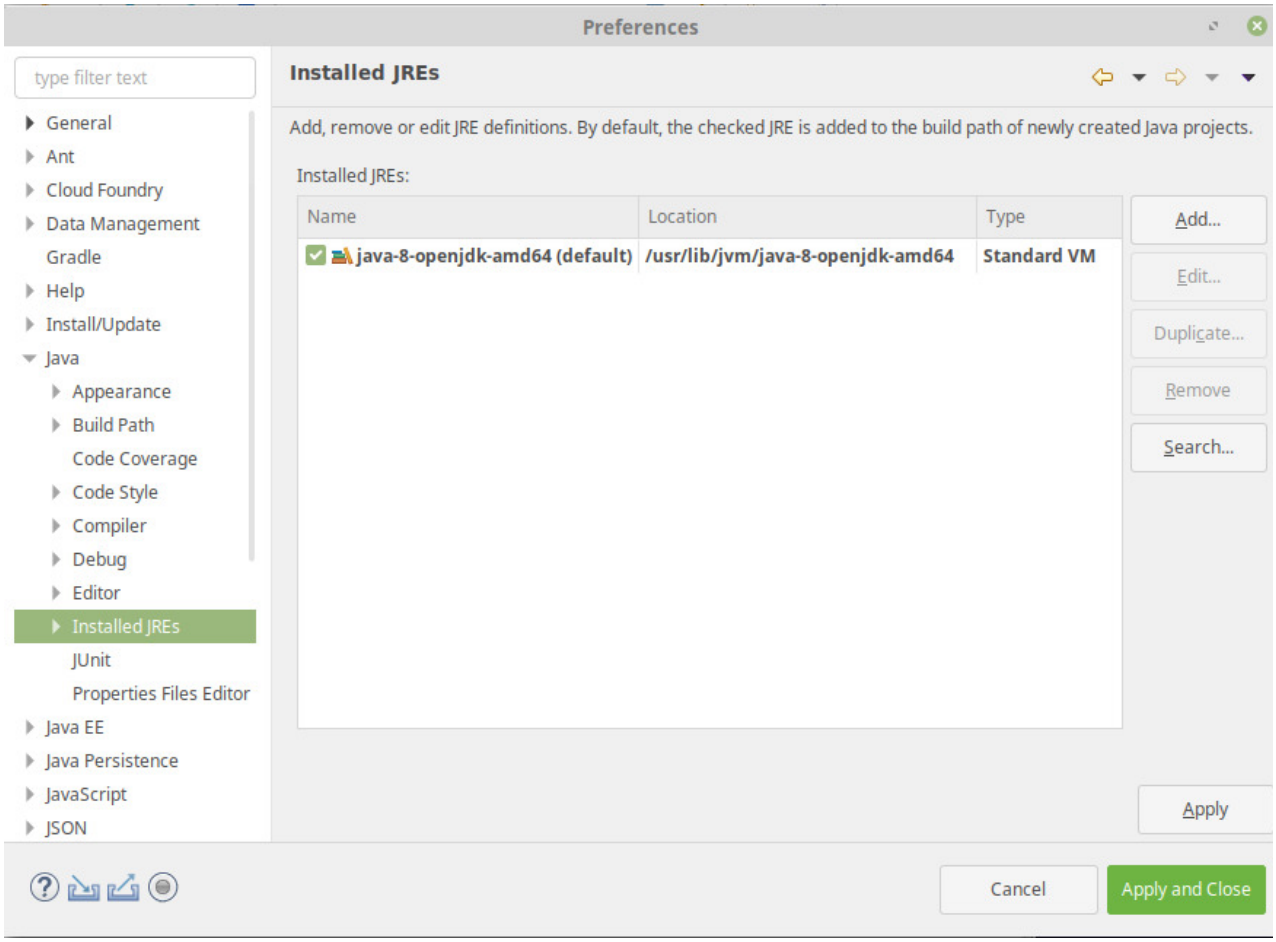
More information at https://eclipse.org/downloads/.

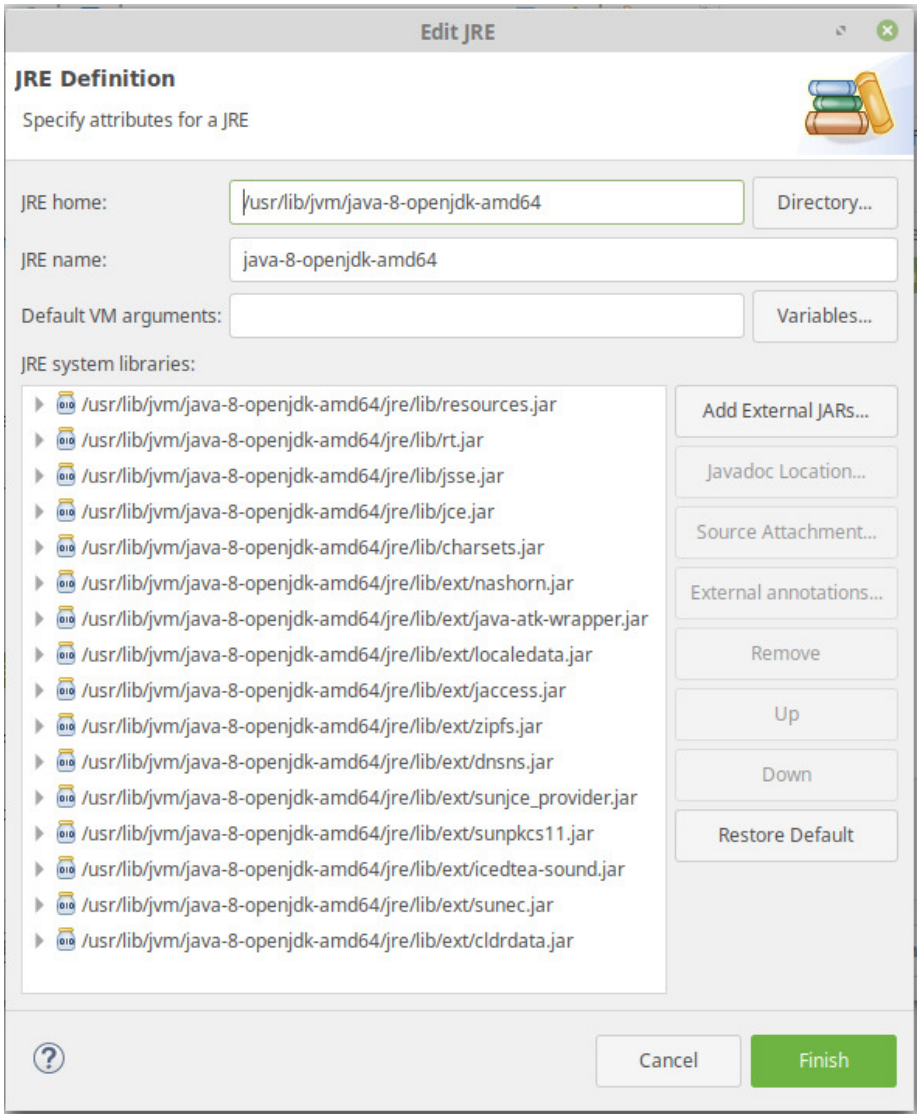
Configure Java

You must have Java installed on your server; otherwise, go to http://www.oracle.com/technetwork/java/index.html, click on Java SE and download the latest OpenJDK 1.8 version according to your OS and hardware architecture. Run the Java installer and follow the installation wizard.

*Go to **Window** / **Preferences** / **Java** / **Installed JRE**. You must have OpenJDK 1.8 defined here:*

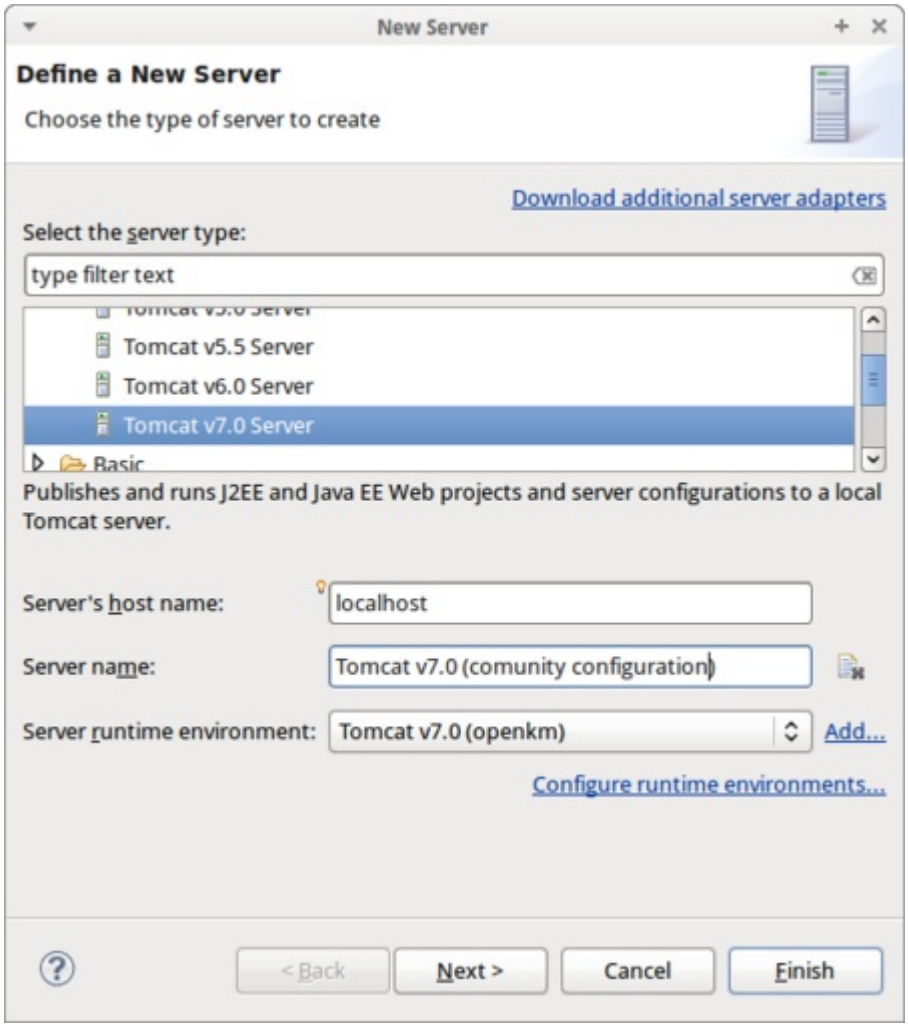
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.


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



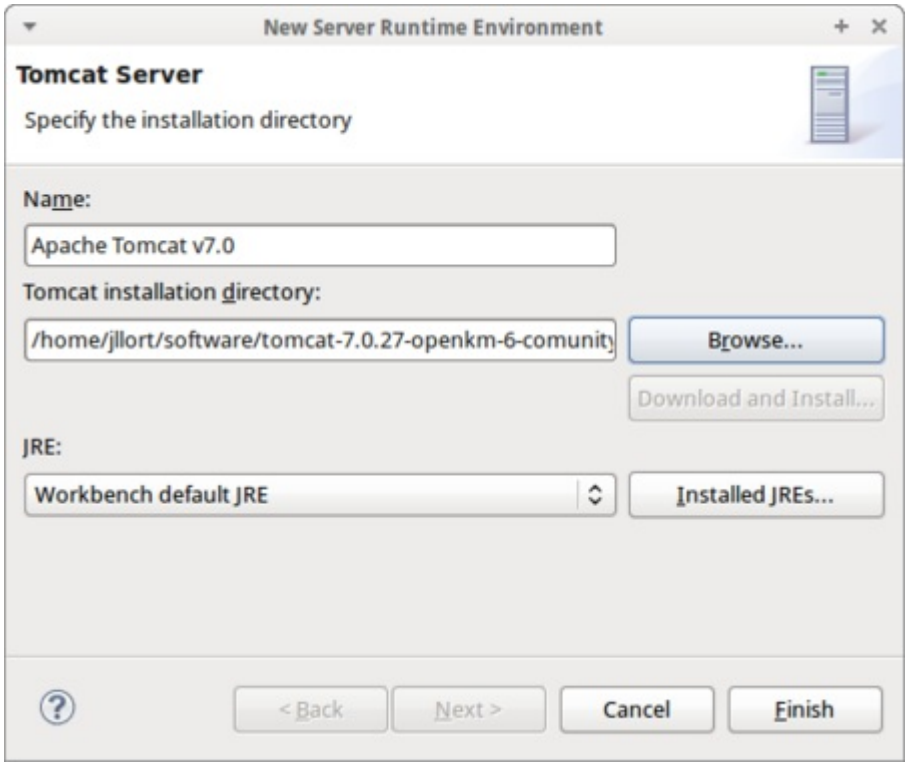
Configure Tomcat server

*Go to the "**Servers**" tab. Right-click and, from the contextual menu, choose "**Add / server**". From the list choose "**Tomcat 7 server**":*



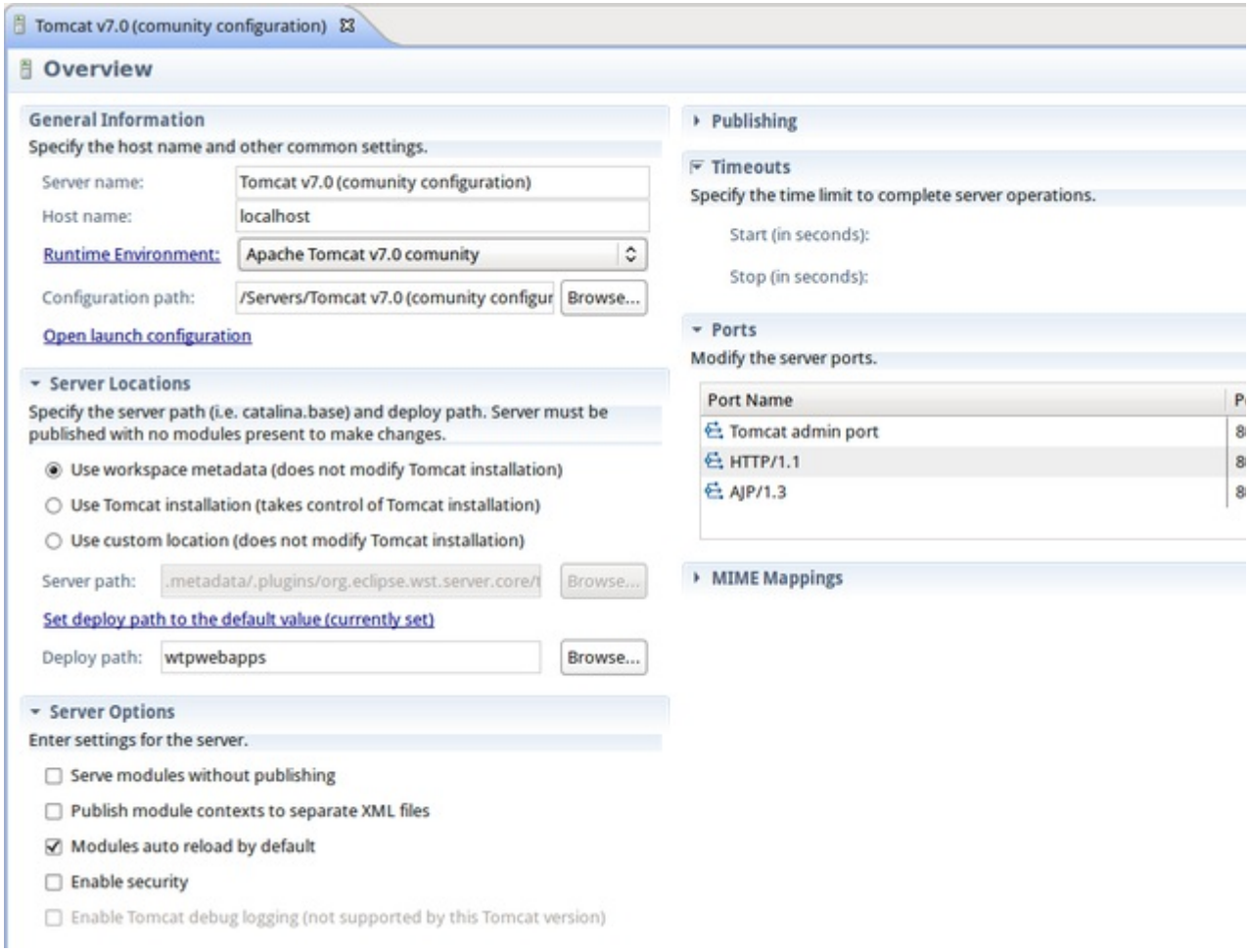
*Configure a new **runtime server environment**:*

> *Choose the "Workbench default JRE" when your default JRE is "OpenJDK 1.8", or choose "OpenJDK 1.8" from the list of installed JREs.*



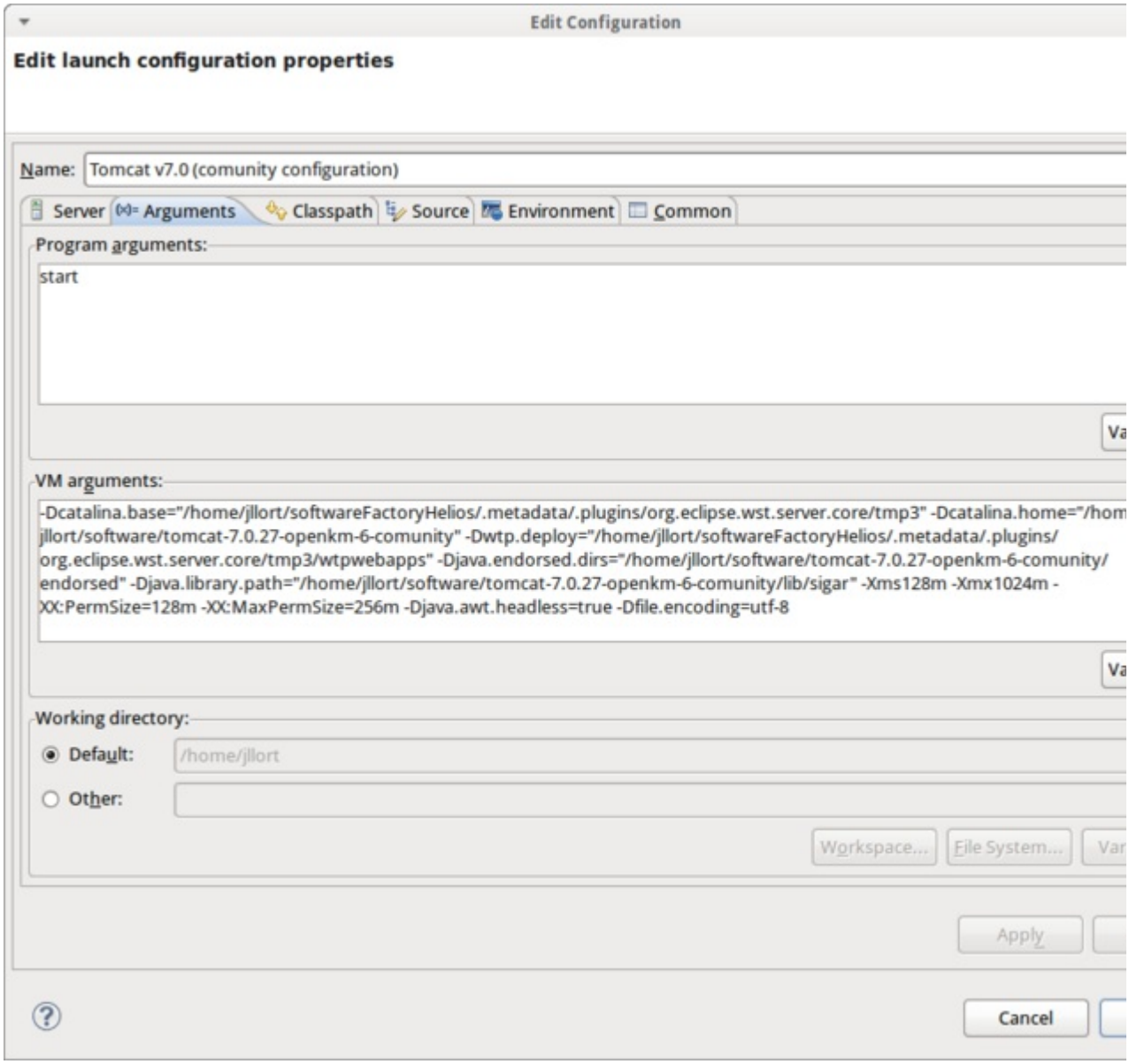
*Check the **Use Tomcat installation** box (takes control of Tomcat installation).*

*Increase the start and stop **timeouts** (for example, to 180 and 120 seconds respectively).*

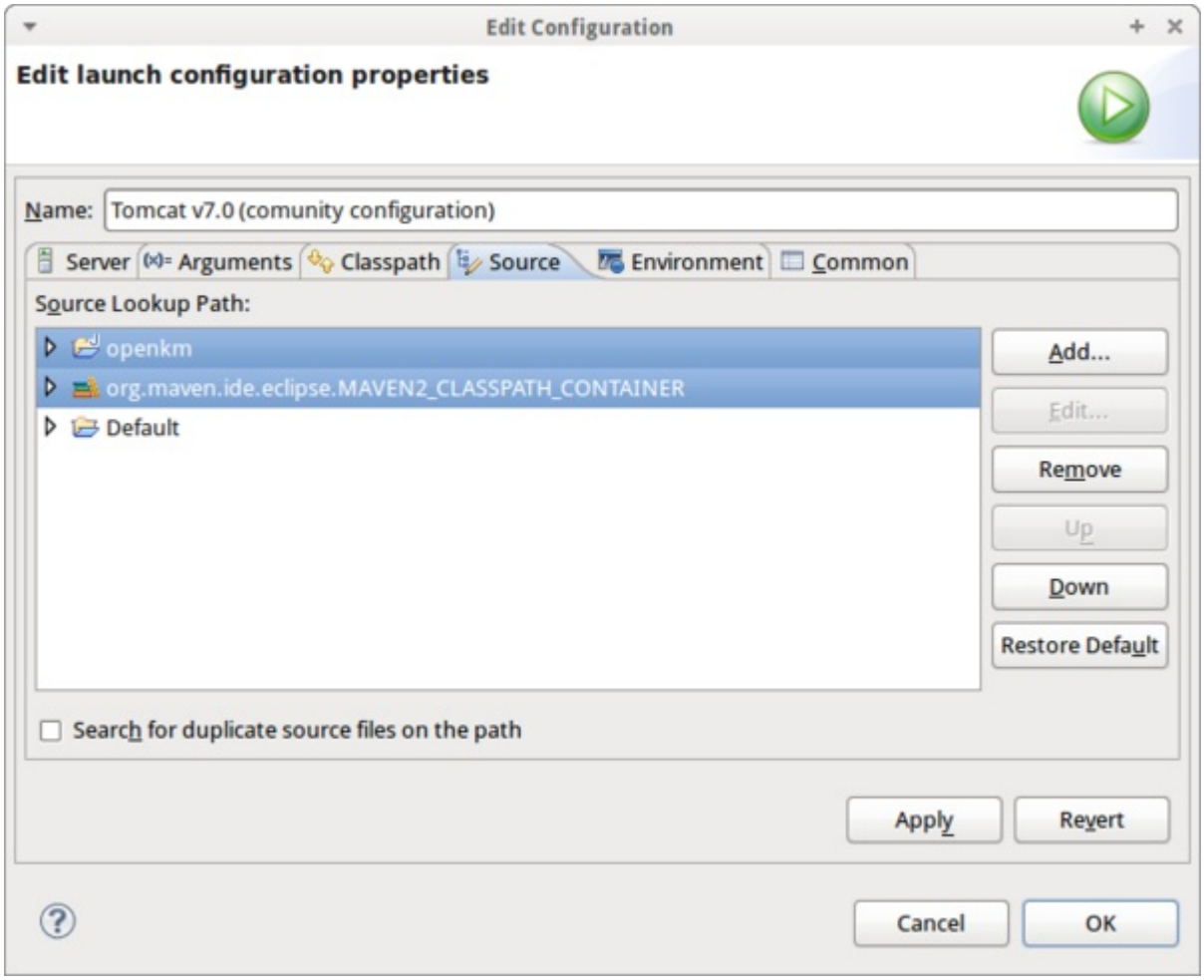


*Edit the **launch configuration**: add **memory parameters** (we suggest 2 GB), the **sigar** library path, and **UTF-8** file encoding in the VM arguments field:*

```
-Xms128m -Xmx2048m -Djava.awt.headless=true -Dfile.encoding=utf-8 -Djava.library.path="/home/openkm/tomcat-
```



*Add your **OpenKM project** on the Source tab:*



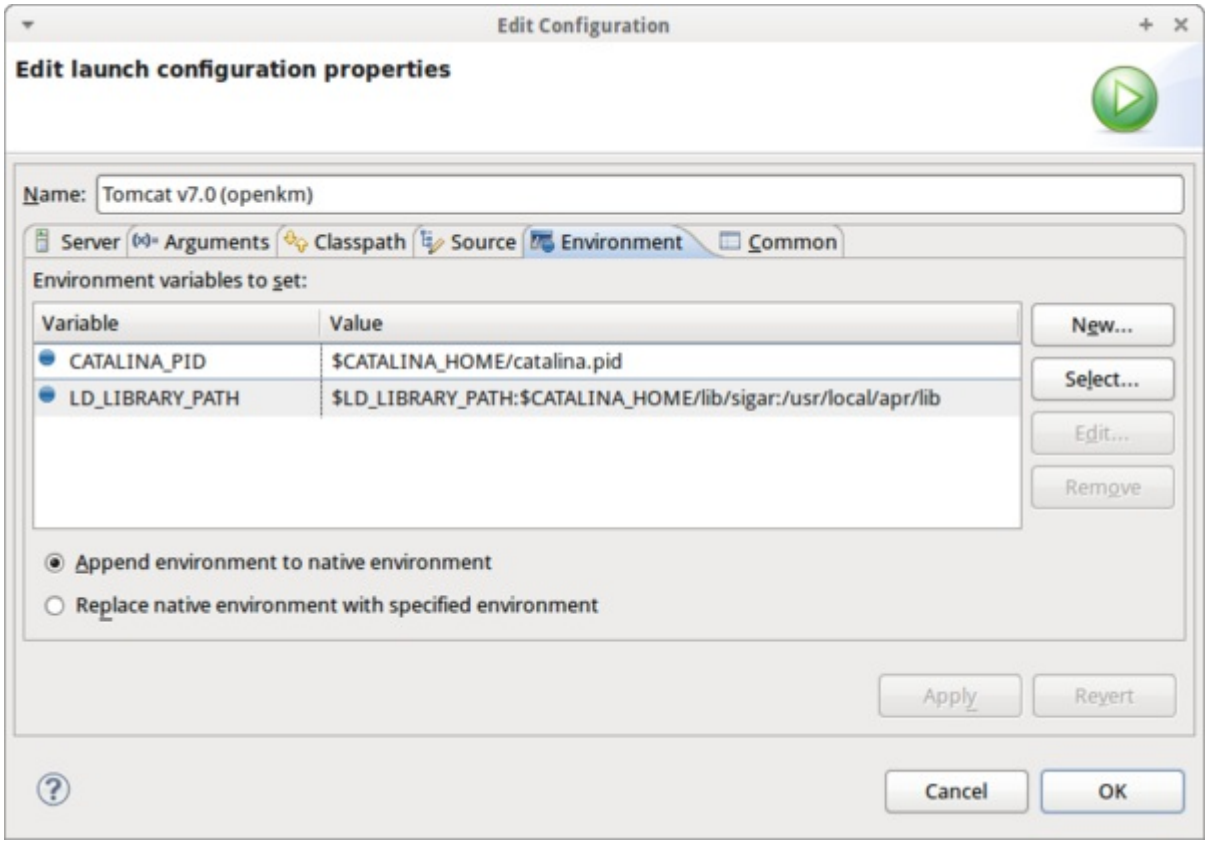
*Add **CATALINA_PID** and **LD_LIBRARY_PATH** to the environment variables:*

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$CATALINA_HOME/lib/sigar:/usr/local/apr/lib
```

or

```
CATALINA_PID=$CATALINA_HOME/catalina.pid
LD_LIBRARY_PATH=/home/openkm/tomcat-8.5.69/lib/sigar
```

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



Maven support for Eclipse

The latest Eclipse version comes by default with Maven support plugins already installed. For older Eclipse versions, follow these configuration steps:

- *Go to **Help** > **Eclipse Marketplace***
- *Install the plugin named M2Eclipse.*

> *Older Eclipse versions come without the "Eclipse Marketplace" feature, and the plugins must be installed manually using the **Help** > **Install New Software** option. We encourage you to use a more recent version of Eclipse.*

Subversion support for Eclipse

- *Go to **Help** > **Eclipse Marketplace***
- *Install the plugin named Subversive.*

> *To check out an SVN project, go to **File** > **New** > **Others** > **Maven** > **Check out Maven Projects from SCM** .*
>
> *At the beginning, the SCM URL list will be empty; you must click the bottom-right "**m2e Marketplace**" link and install the "**m2e-subversive**" connector.*

*Older Eclipse versions come without the "Eclipse Marketplace" feature, and the plugins must be installed manually using the **Help > Install New Software** option. We encourage you to use a more recent version of Eclipse.*

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

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 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, you can do so by executing the following command:

```
$ mvn install -Dmaven.test.skip=true
```

Memory problems

To set more memory for Maven in GNU/Linux, you can add the Maven executable to the binary path 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 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 tries to log important events like 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 a lot of messages. These files are stored at $TOMCAT_HOME/logs.

Debug remote Tomcat server

When you have problems in the production environment, this configuration is especially useful.

Add the following options when the JVM is started:

```
-Xdebug -Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n
```

*Or if you are using shell scripts to start Tomcat, start it with the following command (**recommended**):*

```
$ $TOMCAT_HOME/bin/catalina.sh jpda start
```

It will start Tomcat so that a remote debugger can connect to port 8000.

More information at:

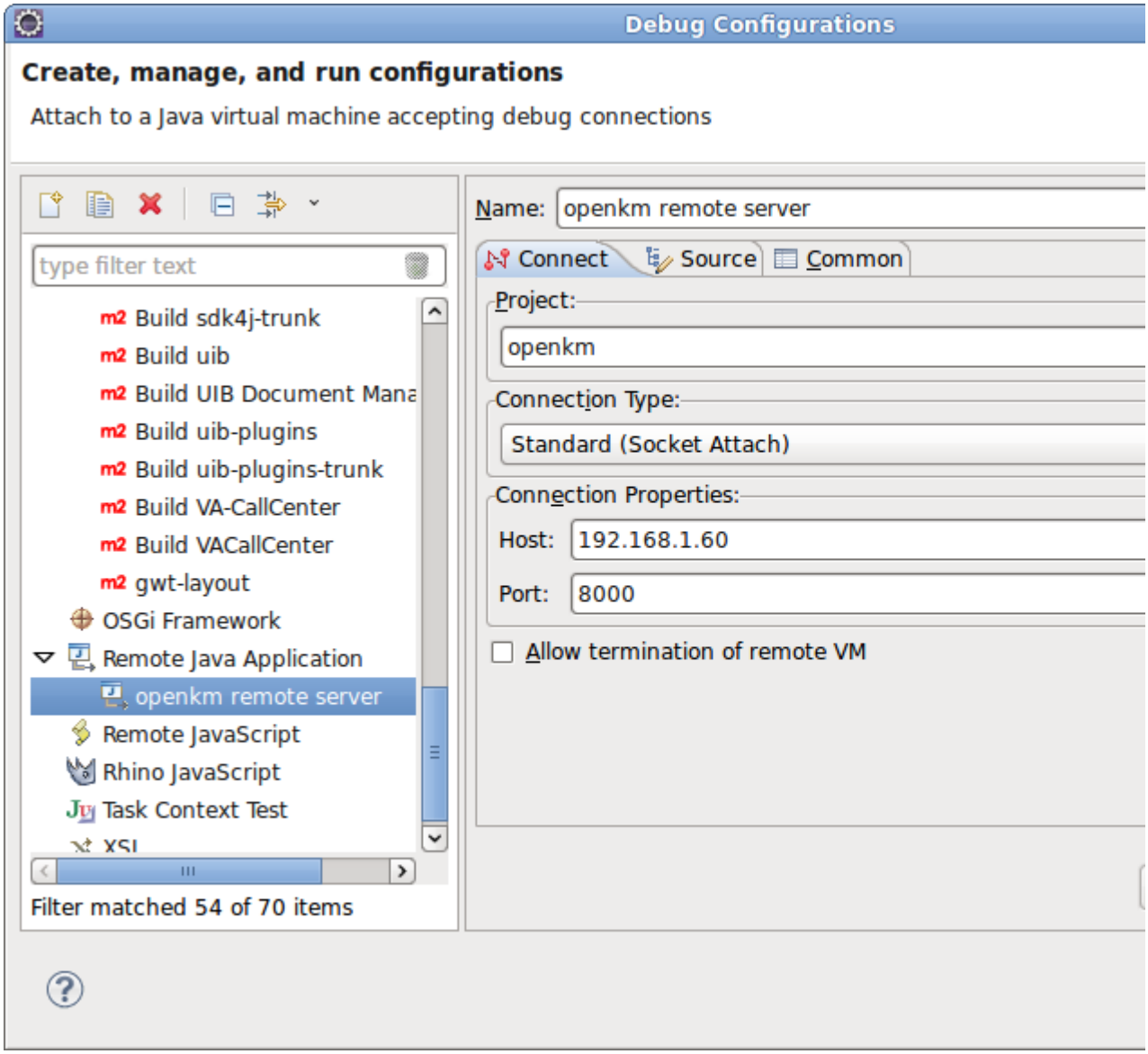
- *Tomcat development: How do I configure Tomcat to support remote debugging?*
- *Como hacer debug remoto en Tomcat desde Eclipse .*

Eclipse configuration

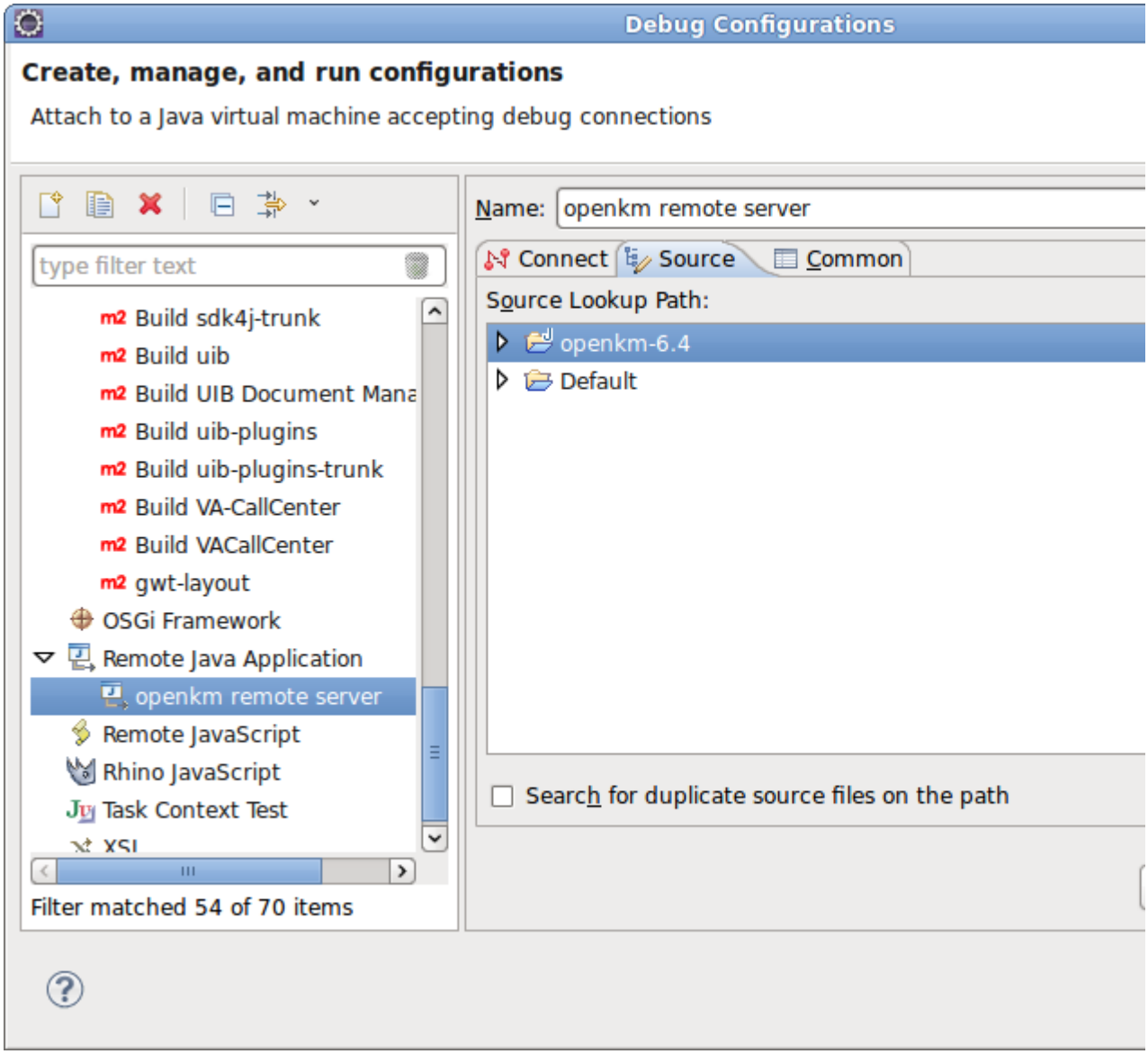
*Open Eclipse "**Debug Configurations**", then choose "**Remote Java Application**", right-click and choose "**New**".*

*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, setting breakpoints, etc.

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 ***CreateProcess 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>
```

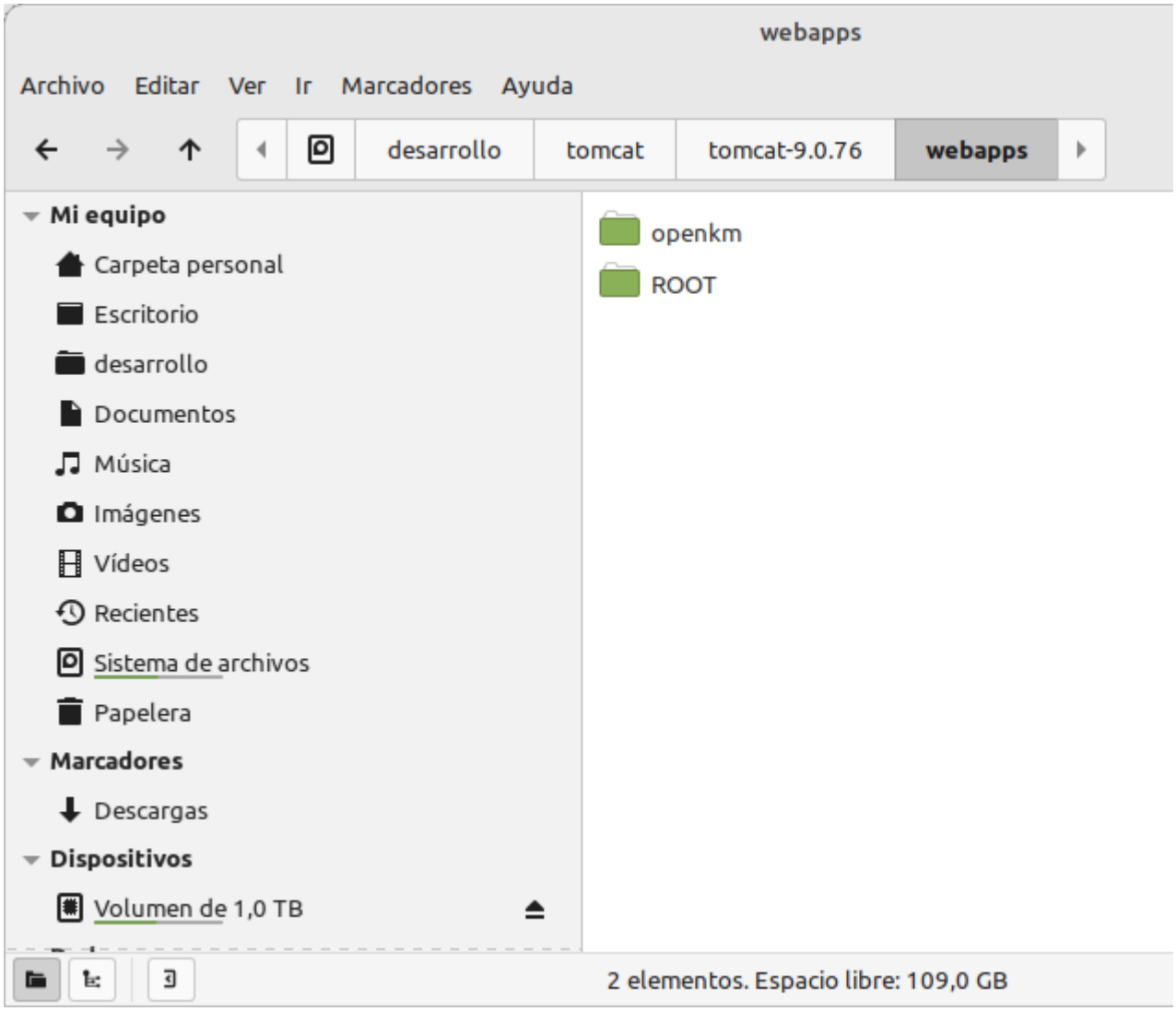
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

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
```

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

Profiling application

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; replace the default **127.0.0.1** value 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.*
> - *Additionally, you can 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 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**:*

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

```
monitorRole tomcat
controlRole tomcat
```

Restrict access to these files:

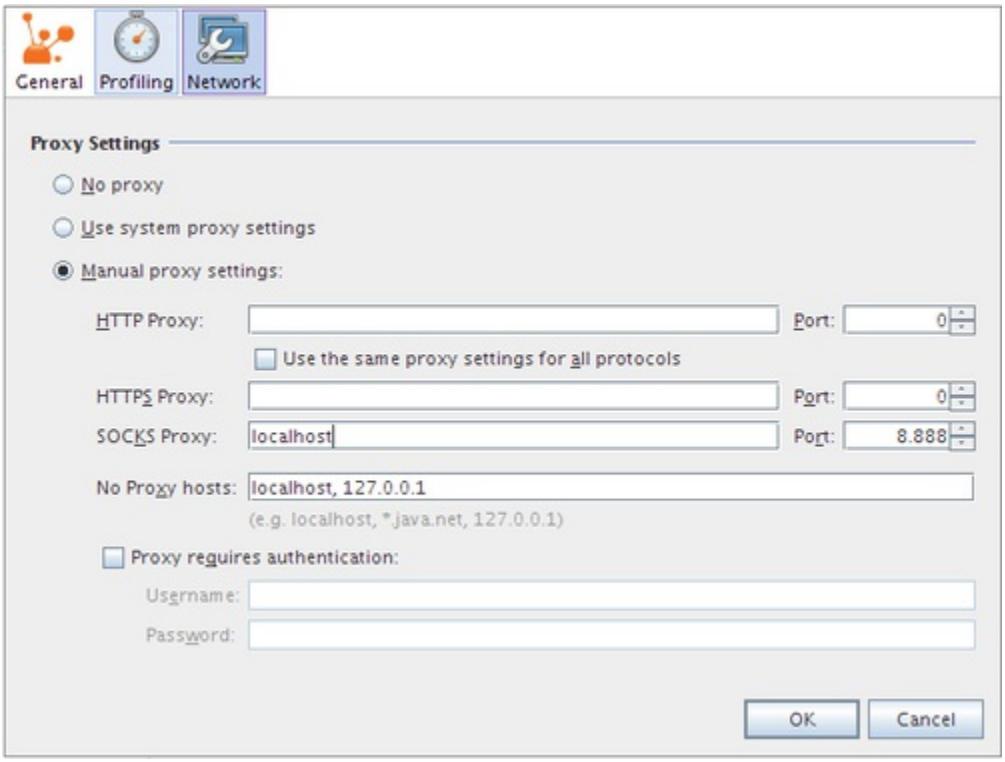
```
$ chmod 600 $CATALINA_HOME/conf/jmxremote.access

$ chmod 600 $CATALINA_HOME/conf/jmxremote.password
```

Monitoring Remote JVM Over SSH

The easiest way to connect to the remote JMX is to use a SOCKS proxy.

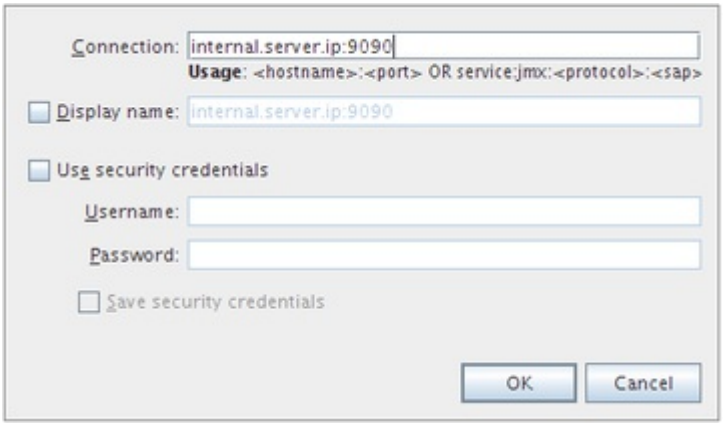
```
$ ssh user@external.server.ip -D 8888
```

Configure VisualVM to use this proxy. Go to ***Tools*** *>* ***Options*** *>* ***Network*** *and edit* ***Manual Proxy Settings****. Configure SOCKS Proxy at localhost and port 8888.*



Now you can connect VisualVM to the remote target. Go to ***File*** *>* ***Add JMX Connection...*** *and type the IP or hostname of the remote machine and the configured JMX port (in our sample configuration it is 9090).*

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



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, you 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*

Application version compatibility table

Application compatibility changes can be caused by:

- *DTD definition changes. Usually, newer application versions support 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 property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.11//EN"
"file:///path/to/property-groups-3.11.dtd">
```

Compatibility tables:

Property Groups	OpenKM
property-groups-3.11.dtd	Since 7.0.0

Report Parameters	OpenKM
report-parameters-2.1.dtd	Since 7.0.0

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.

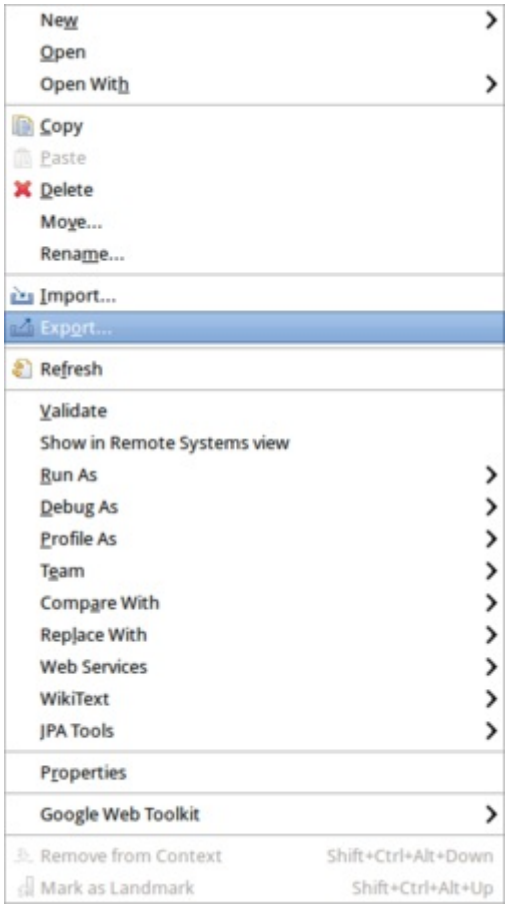
The following steps may change depending on the Eclipse version.

1. ***Choose the Java file*** *from Project Explorer or Navigator view.*
2. ***Right-click*** *(the 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 class of the entry point.*
8. ***Click Finish****.*

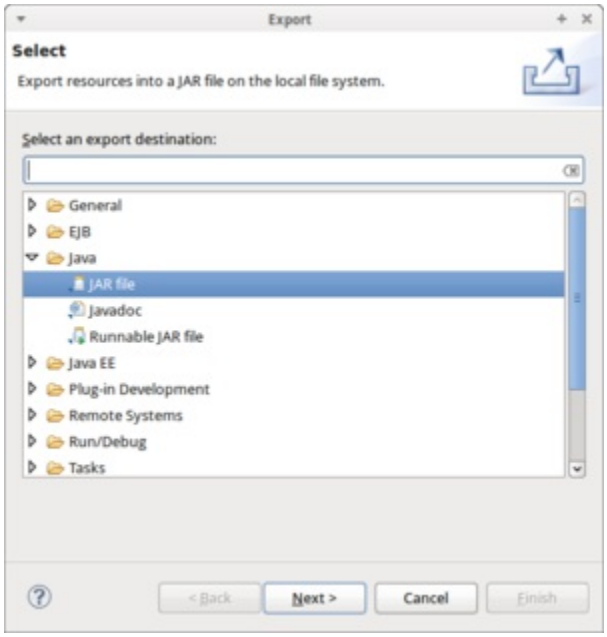
Images

Select the file to be exported as a JAR. Right-click and select the " ***Export****" option.*

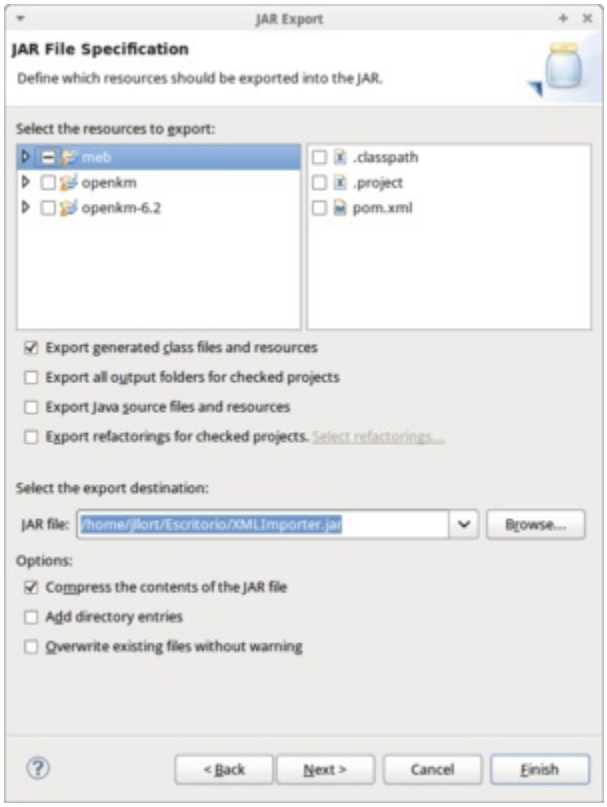
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.



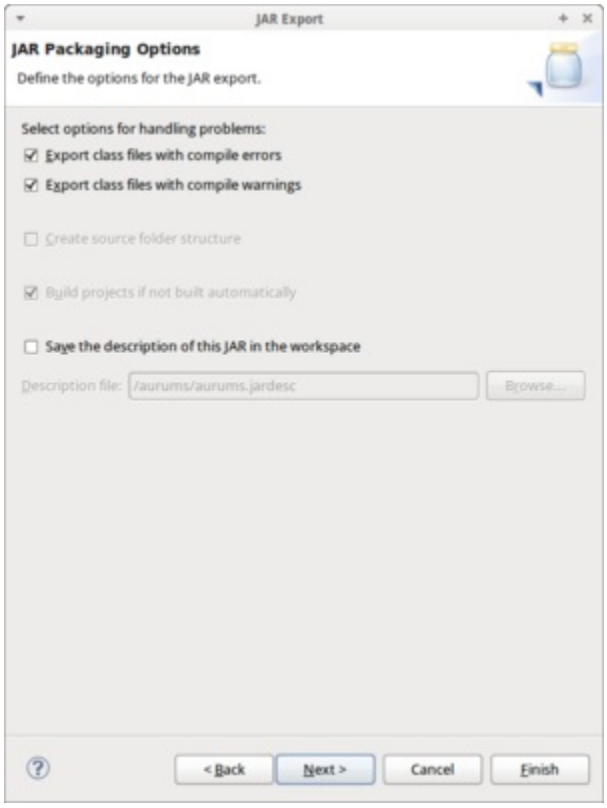
*Select the "**.jar**" file option and click the "**Next**" button.*



*Select the "**Java classes to be included**" to include in the JAR file and click the "**Next**" button.*

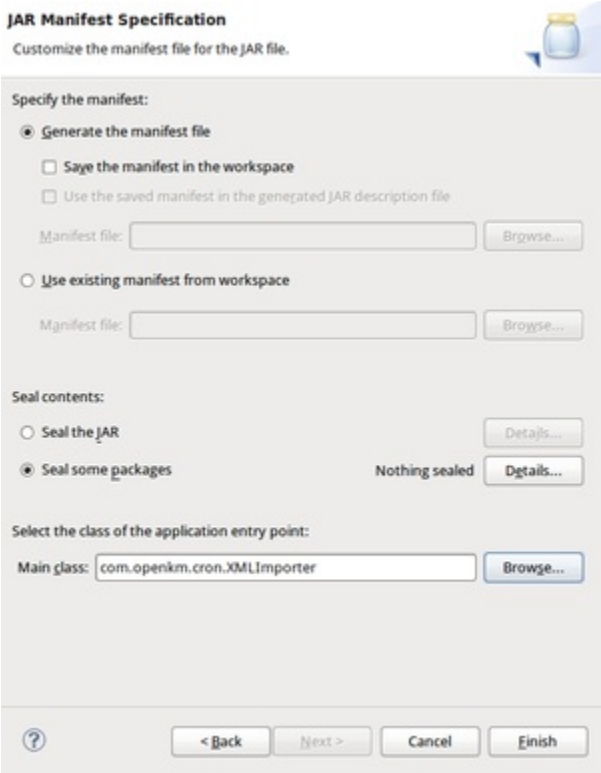


*Click the "**Next**" button.*



*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:

- *The class 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.*

From the OpenKM plugin system, you can:

- *Register new Antivirus analyzers.*
- *Register new Automation actions.*
- *Register new Automation validations.*
- *Register a new cron job.*
- *Register new default form values.*
- *Register a new document converter.*
- *Register a new autocomplete form.*
- *Register new form validators.*
- *Register new NodeProperties.*
- *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 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 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's plugin system will try to load it from there.*

*Do not miss 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*
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";
    }
}
```

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

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 Register a new plugin.

Methods description

Method	*Type*	*Description*
executePre(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, in the "pre" stage.*
executePost(Map<String, Object> env, Object... params)	*void*	*The method executed by the Automation event when the validation conditions succeed, in 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 to get familiar with the AutomationUtils class because the methods to retrieve data are centralized.*
>
> *Take a look at Variables by automation event to see what 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 the Automation.PARAM_TYPE_EMPTY value).

The array can contain 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.*

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 (object mark recognition) will be required.*

Description of available Src values:

Property	*Description*

PARAM_SOURCE_EMPTY	*Indicates an empty source.*
PARAM_SOURCE_FOLDER	*Indicates that the source must be a folder.*
PARAM_SOURCE_OMR	*Indicates that the source must be a valid OMR (object mark recognition).*

Detecting recursion

When you create your automation actions, you should consider recursion. 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 the **action** is set to also **create a new document**.*
- *Automation task linked with "**Set metadata group**" event when the **action** is set to also **change metadata group values**.*

Complex recursive sample:

- *Automation task linked with "**Create document**" event when the **action** is set to **change metadata group values**, and an automation task linked with the "**Set metadata group**" event when the **action** is set to **create a new document**.*

Code to detect recursive calls caused by automation action classes:

```
if (StackTraceUtils.isCallingMe(this.getClass().getName())) {
  log.info("Recursion detected");
  return;
}
```

Example

***AddKeyword** class:*

```
package com.openkm.plugin.automation.action;

import com.openkm.core.Config;
import com.openkm.db.bean.Automation;
import com.openkm.db.bean.AutomationRule.EnumEvents;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.plugin.BasePlugin;
```

```
import com.openkm.plugin.automation.Action;
import com.openkm.plugin.automation.AutomationException;
import com.openkm.plugin.automation.AutomationUtils;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

/**
 * AddKeyword
 */
@PluginImplementation
public class AddKeyword extends BasePlugin implements Action {

    private static ArrayList<EnumEvents> EVENTS_AT_PRE = new ArrayList<>();

    private static ArrayList<EnumEvents> EVENTS_AT_POST
            = Stream.of(EnumEvents.EVENT_DOCUMENT_CREATE, EnumEvents.EVENT_DOCUMENT_UPDATE, E
                    EnumEvents.EVENT_DOCUMENT_RENAME, EnumEvents.EVENT_DOCUMENT_MOVE, EnumEven
                    EnumEvents.EVENT_DOCUMENT_DOWNLOAD_FROM_UI, EnumEvents.EVENT_DOCUMENT_DO
                    EnumEvents.EVENT_DOCUMENT_PRINT, EnumEvents.EVENT_DOCUMENT_STAMP, EnumEvents
                    EnumEvents.EVENT_RECORD_DELETE, EnumEvents.EVENT_FOLDER_CREATE, EnumEvents.EV
                    EnumEvents.EVENT_MAIL_CREATE, EnumEvents.EVENT_MAIL_DOWNLOAD_FROM_UI, EnumEve
                    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 in 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 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 descriptions

Method	*Type*	*Description*
isValid(Map<String, Object> env, Object... params)	*boolean*	*The method executed by the automation event evaluates whether conditions are true or false.*
getName()	*String*	*Sets the name that will be shown in the administrator user interface selector list.*
getParamType00 ***getParamType01*** ***getParamType02***	*String*	*Sets the parameter type.* *Available values:* • *Automation.PARAM_TYPE_EMPTY* • *Automation.PARAM_TYPE_TEXT* • *Automation.PARAM_TYPE_INTEGER* • *Automation.PARAM_TYPE_BOOLEAN* • *Automation.PARAM_TYPE_TEXTAREA* • *Automation.PARAM_TYPE_CODE* • *Automation.PARAM_TYPE_USER* • *Automation.PARAM_TYPE_ROLE* • *Automation.PARAM_TYPE_OMR* *When the value is set to Automation.PARAM_TYPE_EMPTY.* *The parameters work as a logical group.* ***[getParamType00 + getParamSrc00 + getParamDesc00]***
getParamSrc00	*String*	*Sets the source type.*

getParamSrc01 ***getParamSrc02***		*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 the env variable

The env variable present in the isValid method 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 become familiar with the AutomationUtils class, because the methods for retrieving data are centralized there.*
>
> *Take a look at Variables by automation event to see what 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 present in the isValid method is an array of Objects. This array is filled by the automation event based on the number of parameters set in the Validation class (which will not be the Automation.PARAM_TYPE_EMPTY value).

The array can contain several objects of distinct types; it is necessary to retrieve the parameters as the correct object type.

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

It takes some time to become 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.*

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, EnumEve
                    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, EnumEve
                    EnumEvents.EVENT_TEXT_EXTRACTOR, EnumEvents.EVENT_PROPERTY_GROUP_ADD, EnumE
                    EnumEvents.EVENT_PROPERTY_GROUP_REMOVE, EnumEvents.EVENT_MAIL_IMPORT_ATTAC
            ).collect(Collectors.toCollection(ArrayList::new));

    @Autowired
    private AutomationUtils automationUtils;
```

```
@Autowired
private OKMRepository okmRepository;

@Autowired
private NodeBaseSrv nodeBaseSrv;

@Override
public boolean isValid(Map<String, Object> env, Object... params) throws AutomationException {
    try {
        String fldUuid = automationUtils.getString(0, params);
        NodeBase node = automationUtils.getNodeToEvaluate(env);
        String parentPath = nodeBaseSrv.getPathFromUuid(node.getParent());
        String fldPath = okmRepository.getNodePath(null, fldUuid);

        if (parentPath.startsWith(fldPath)) {
            return true;
        } else {
            return false;
        }
    } catch (Exception e) {
        throw new AutomationException("PathContains exception", e);
    }
}

@Override
public String getName() {
    return "PathContains";
}

@Override
public String getParamType00() {
    return Automation.PARAM_TYPE_TEXT;
}

@Override
public String getParamSrc00() {
    return Automation.PARAM_SOURCE_FOLDER;
}

@Override
public String getParamDesc00() {
    return "String";
}

@Override
public String getParamType01() {
    return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc01() {
    return Automation.PARAM_SOURCE_EMPTY;
}

@Override
public String getParamDesc01() {
    return "";
}

@Override
public String getParamType02() {
    return Automation.PARAM_TYPE_EMPTY;
}

@Override
public String getParamSrc02() {
    return Automation.PARAM_SOURCE_EMPTY;
```

```
    }

    @Override
    public String getParamDesc02() {
        return "";
    }

    @Override
    public List<EnumEvents> getValidEventsAtPre() {
        return EVENTS_AT_PRE;
    }

    @Override
    public List<EnumEvents> getValidEventsAtPost() {
        return EVENTS_AT_POST;
    }
}
```

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

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's 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
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 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* ***token value of null****. That indicates the application must use the logged-in session.*

When ***a user is not logged in****, the call to the API must use a* ***non-null token****; otherwise, an error will occur.*

Example of use:

- *logged-in user -> **valid** API call -> OKMFolder.getInstance().createSimple(null, "/okm:root/test");*
- *not logged-in user -> **invalid** API call -> OKMFolder.getInstance().createSimple(**null**, "/okm:root/test");*

Using the system token is a good practice when calling API methods in crontab tasks:

```
String systemToken = DbSessionManager.getInstance().getSystemsystemToken();
OKMFolder.getInstance().createSimple(systemToken, "/okm:root/test")
```

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

Creating your own Autocomplete Form plugin

Available from the OpenKM version 7.1.41 and the DTD version 3.11

You can create your own autocomplete form.

Conditions:

- *The new Default Form Values class* ***must implement the "AutocompleteFormValues"*** *interface.*
- *The new Default Form Values class* ***must be declared 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 plugin system will try to load it from there. See the sample below:*

Do not miss the tag ***@PluginImplementation****, otherwise the application plugin system will not be able to retrieve the new class.*

More information at: Register a new plugin

Method description

Method	*Type*	*Description*
autocompleteFormValues(String uuid, String grpName, List<FormElement> formElementList)	*void*	*The method autocompletes the form elements of the group.*

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 3.10//EN"
    "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="Autocomple sample" name="okg:autocomplete" autocompleteValueClassName="com.ope
        <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>
```

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

Creating your own Default Form Values

Starting with OpenKM version 7.1.38, the method was changed to include 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 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 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 plugin system will try to load it from there. See the sample below:*

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*

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

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/pr
<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>
```

Creating your own Document Converter

You can create your own Document Converter.

Conditions:

- *The new Document Converter class **must implement the "Converter"** interface.*
- *The new Document Converter class **must be declared under the package "com.openkm.plugin.conversion".***
- *The new Document Converter class **must be annotated with "@PluginImplementation"**.*
- *The new Document Converter class **must extend 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 allowed to be converted.*
getConversionType()	*int*	*Returns the MIME type value 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 a 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 by server-side logic 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 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 nodes affected (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 in the metadata field named okp:technology.comment will not be allowed.

```
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

This method is executed when adding, updating, and checking if a node can be promoted as a record.

The application checks if a node can be promoted as a record when the file plan configuration is enabled.

The variable fullValidation is set to true only when the application is checking if a node can be promoted as a record.

Form validators are used to implement complex logic that must be evaluated by server-side logic 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 Valio

 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 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, it should throw an exception of type ValidationFormException.* *The node variable is the OpenKM node where the user added or updated the metadata.* *The grpName is the name of the metadata group.* *The properties variable contains a map with all the metadata values set by the user.* *The variable fullValidation is set to true when checking if a node can be promoted as a record.*

Example of the Form validator implementation

Metadata definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
        "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
    <property-group label="Full validation" name="okg:full" validatorClassName="com.openkm.plugin.form.validator.F
        <input label="First number" name="okp:full.number1"></input>
        <input label="Second number" name="okp:full.number2"></input>
        <input label="Third number" 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 NodeProperties plugin (customizing search result objects)

NodeProperties is usually used in Web services scenarios. Using it, you can customize the result objects returned by 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 better performance (less time for filling data on the server side and less time for marshalling and unmarshalling in the web services procedure).

You can create your own NodeProperties.

Conditions:

- *The new NodeProperties class* ***must implement the "NodeProperties"*** *interface.*
- *The new NodeProperties class* ***must be declared under the package "com.openkm.plugin.properties".***
- *The new NodeProperties class* ***must be annotated with "@PluginImplementation".***
- *The new NodeProperties class* ***must extend "BasePlugin".***

NodeProperties interface:

```
package com.openkm.plugin.properties;

import com.openkm.bean.Document;
import com.openkm.bean.Folder;
import com.openkm.bean.Mail;
import com.openkm.bean.Record;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.core.RepositoryException;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import net.xeoh.plugins.base.Plugin;

/**
 * NodeProperties
 */
public interface NodeProperties extends Plugin {

    /**
     * Get document properties
     */
    Document getProperties(String user, NodeDocument nDocument, String linkTarget) throws PathNotFoundExcept

    /**
     * Get folder properties
     */
    Folder getProperties(String user, NodeFolder nFolder) throws PathNotFoundException, DatabaseException, Rep

    /**
     * Get mail properties
     */
    Mail getProperties(String user, NodeMail nMail) throws PathNotFoundException, DatabaseException, Repository

    /**
```

```
     * Get record properties
     */
    Record getProperties(String user, NodeRecord nRecord) throws PathNotFoundException, DatabaseException, R
}
```

*The **new class must be loaded into the package com.openkm.plugin.properties** because the application plugin system will try to load it from there.*

> *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
getProperties(String user, NodeDocument nDocument, String linkTarget)	**Document**	Define which values are loaded in Document Object returned by search engine.
getProperties(String user, NodeFolder nFolder)	**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 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 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.*
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.CommonUser;
import com.openkm.bean.form.Option;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;
import java.util.List;

@PluginImplementation
public class OptionSelectUserList extends BasePlugin implements OptionSelectValues {

    @Autowired
    private OKMAuth okmAuth;

    @Override
    public String getName() {
        return "User list";
    }

    @Override
    public List<Option> getOptions() throws OptionSelectException {
        List<Option> options = new ArrayList<>();

        try {
            List<CommonUser> userList = okmAuth.getUsers(null);
            for (CommonUser user : userList) {
                Option option = new Option();
                option.setLabel(user.getName());
                option.setValue(user.getId());
                options.add(option);
            }
        } catch (PrincipalAdapterException e) {
            throw new OptionSelectException(e);
        }

        return options;
    }
}
```

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

Creating your own Principal Adapter

Principal adapters are used by the authentication module to retrieve user and role information. For example, to retrieve users and roles from an LDAP directory or an external database that stores 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 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.

*To **enable the new Principal adapter**, go to **Administration > Configuration parameters** > find the parameter named **principal.adapter** and set the 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 a list of all users.
getRoles()	List<String>	Returns a list of all roles.
getUser(String userId)	CommonUser	Returns all user data.
getUsersByRole(String role)	List<CommonUser>	Returns a list of all users who have been assigned a role.
getRolesByUser(String user)	List<String>	Returns a list of all roles assigned to a user.
String getPassword(String user)	String	Returns the password associated with a specific user.

createUser(CommonUser user)	*void*	*Creates a new user.*
deleteUser(String user)	*void*	*Deletes a user.*
updateUser(CommonUser user)	*void*	*Updates a user.* *Password parameter can be null or empty.*
createRole(String role, boolean active)	*void*	*Creates a new role.*
deleteRole(String role)	*void*	*Deletes a role.*
updateRole(String role, boolean active)	*void*	*Updates a role.*
assignRole(String user, String role)	*void*	*Assigns a role to a user.*
removeRole(String user, String role)	*void*	*Revokes a role from a user.*
getProfiles(boolean filterByActive)	*List<Profile>*	*Returns a list of all profiles.*
getUserProfile(String userId)	*Profile*	*Returns the profile assigned to a user.*
setUserProfile(String userId, long profileId)	*void*	*Changes the profile assigned to a user.*
isManageUsers()	*boolean*	*Returns a boolean that indicates whether the user is a manager.*
isManageRoles()	*boolean*	*Returns a boolean that indicates whether the role is managed.*

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

Example of the Principal Adapter implementation

The example retrieves users, roles, and emails from properties files.

```
package com.openkm.plugin.principal;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Enumeration;
import java.util.List;
import java.util.Properties;
import java.util.StringTokenizer;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.core.Config;
import com.openkm.db.bean.Profile;
import com.openkm.plugin.BasePlugin;
import com.openkm.principal.PrincipalAdapterException;

public class UsersRolesPrincipalAdapter extends BasePlugin implements PrincipalAdapter {

    private static Logger log = LoggerFactory.getLogger(UsersRolesPrincipalAdapter.class);

    @Autowired
    private OKMAuth okmAuth;

    @Override
    public List<CommonUser> getUsers() throws PrincipalAdapterException {
        log.debug("getUsers()");
        List<CommonUser> list = new ArrayList<>();
        Properties prop = new Properties();

        try {
            prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-users.properties"));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        for (Enumeration<Object> e = prop.keys(); e.hasMoreElements();) {
            String user = (String) e.nextElement();
            if (!Config.SYSTEM_USER.equals(user)) {
                CommonUser commonUser = okmAuth.getUser(null, user);
                list.add(commonUser);
            }
        }

        log.debug("getUsers: {}", list);
        return list;
    }

    @Override
    public List<String> getRoles() throws PrincipalAdapterException {
        log.debug("getRoles()");
        List<String> list = new ArrayList<String>();
        Properties prop = new Properties();
```

```
        try {
            prop.load(new FileInputStream(Config.HOME_DIR + "/server/default/conf/props/openkm-roles.properties"));
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        for (Enumeration<Object> e = prop.elements(); e.hasMoreElements();) {
            for (StringTokenizer st = new StringTokenizer((String) e.nextElement(), ","); st.hasMoreTokens();) {
                String role = st.nextToken();

                if (!Config.DEFAULT_ADMIN_ROLE.equals(role) && !list.contains(role)) {
                    list.add(role);
                }
            }
        }

        log.debug("getRoles: {}", list);
        return list;
    }

    @Override
    public CommonUser getUser(String userId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getUser");
    }

    @Override
    public List<CommonUser> getUsersByRole(String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getUsersByRole");
    }

    @Override
    public List<String> getRolesByUser(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getRolesByUser");
    }

    @Override
    public String getPassword(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getPassword");
    }

    @Override
    public CommonUser createUser(CommonUser user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("createUser");
    }

    @Override
    public void deleteUser(String user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("deleteUser");
    }

    @Override
    public CommonUser updateUser(CommonUser user) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("updateUser");
    }

    @Override
    public void createRole(String role, boolean active) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("createRole");
    }

    @Override
    public void deleteRole(String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("deleteRole");
    }
```

```
    @Override
    public void updateRole(String role, boolean active) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("updateRole");
    }

    @Override
    public void assignRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("assignRole");
    }

    @Override
    public void removeRole(String user, String role) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("removeRole");
    }

    @Override
    public List<Profile> getProfiles(boolean filterByActive) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getProfiles");
    }

    @Override
    public Profile getUserProfile(String userId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("getUserProfile");
    }

    @Override
    public void setUserProfile(String userId, long profileId) throws PrincipalAdapterException {
        throw new UnsupportedOperationException("setUserProfile");
    }

    @Override
    public boolean isManageUsers() {
        throw new UnsupportedOperationException("isManageUsers");
    }

    @Override
    public boolean isManageRoles() {
        throw new UnsupportedOperationException("isManageRoles");
    }

}
```

Creating your own 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 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****.*

To make this change 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 listed (more information in the class Permission.java):

0 NONE
1 READ
2 WRITE
4 DELETE
8 SECURITY
16 MOVE
1024 DOWNLOAD
2048 not used
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 session user has the grant.*
isGranted(NodeBase node, String user, int permissions)	***Boolean***	*Returns true when the user has the grant.*
isGranted(NodeBase node, long nodeClass, int permissions)	***Boolean***	*Returns true when the session user has the grant.* *This method only applies when the file plan is enabled.*

Example of a security access manager implementation

```
package com.openkm.plugin.access;

import com.openkm.core.AccessDeniedException;
import com.openkm.core.Config;
import com.openkm.core.DatabaseException;
import com.openkm.core.PathNotFoundException;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeClass;
import com.openkm.db.service.NodeBaseSrv;
import com.openkm.db.service.NodeClassSrv;
import com.openkm.module.common.CommonAuthModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.plugin.fileplan.security.FilePlanAccessManager;
import com.openkm.principal.PrincipalAdapterException;
import com.openkm.principal.PrincipalUtils;
import com.openkm.util.StackTraceUtils;
```

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

```
import net.xeoh.plugins.base.annotations.PluginImplementation;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

/**
 * Check user permissions on documents and folders.
 *
 * @see com.openkm.plugin.access.DbRecursiveAccessManager
 */
@PluginImplementation
public class DbSimpleAccessManager extends BasePlugin implements DbAccessManager {
 private static final Logger log = LoggerFactory.getLogger(DbSimpleAccessManager.class);

 @Autowired
 private CommonAuthModule commonAuthModule;

 @Autowired
 private NodeClassSrv nodeClassSrv;

 @Autowired
 private NodeBaseSrv nodeBaseSrv;

 /**
  * Check for permissions.
  */
 public void checkPermission(NodeBase node, int permissions) throws AccessDeniedException, PathNotFoundExce
  if (!isGranted(node, permissions)) {
   String nodePath = nodeBaseSrv.getPathFromUuid(node.getUuid());
   throw new AccessDeniedException(node.getUuid() + " : " + nodePath);
  }
 }

 /**
  * Check for permissions.
  */
 @Override
 public boolean isGranted(NodeBase node, int permissions) throws DatabaseException {
  String userId = PrincipalUtils.getUser();
  Set<String> roles = PrincipalUtils.getRoles();
  return isGranted(node, userId, roles, permissions);
 }

 /**
  * Check for permissions.
  */
 @Override
 public boolean isGranted(NodeBase node, String user, int permissions) throws PrincipalAdapterException, Databas
  List<String> roles = commonAuthModule.getRolesByUser(user);
  return isGranted(node, user, new HashSet<>(roles), permissions);
 }

 @Override
 public boolean isGranted(NodeBase node, long nodeClass, int permissions) throws DatabaseException {
  NodeClass nc = nodeClassSrv.findByPk(nodeClass);
  FilePlanAccessManager accessManager = nodeClassSrv.findAccessManagerByClassName(nc.getAccessManage
  return accessManager.isGranted(node, nodeClass, PrincipalUtils.getUser(), PrincipalUtils.getRoles(), permissions)
 }

 /**
  * Check for permissions.
  */
```

```
private boolean isGranted(NodeBase node, String user, Set<String> roles, int permissions) {
 log.debug("isGranted({}, {})", node.getUuid(), permissions);
 boolean access;

 if (user != null) {
  if (Config.SYSTEM_USER.equals(user) || Config.ADMIN_USER.equals(user)) {
   // An okmAdmin user has total access
   access = true;
  } else {
   if (roles.contains(Config.DEFAULT_ADMIN_ROLE)) {
    // An user with AdminRole has total access
    access = true;
   } else {
    access = checkProperties(node.getUserPermissions(), node.getRolePermissions(), user, roles, permissions);
   }
  }
 } else {
  access = true;

  log.info("*******************************************************");
  StackTraceUtils.logTrace(log);
  log.info("*******************************************************");
 }

 log.debug("isGranted: {}", access);
 return access;
}

/**
 * Check access properties
 */
private boolean checkProperties(Map<String, Integer> usersPerms, Map<String, Integer> rolesPerms, String user,
  Set<String> roles, int perms) {
 log.debug("checkProperties({}, {}, {}, {})", usersPerms, rolesPerms, roles, perms);
 boolean access = false;

 // Fist try with user permissions
 Integer userPerms = usersPerms.get(user);

 if (userPerms != null && (perms & userPerms) != 0) {
  log.debug("checkProperties: {}", true);
  return true;
 }

 // If there is no user specific access, try with roles permissions
 for (String role : roles) {
  Integer rolePerms = rolesPerms.get(role);

  if (rolePerms != null && (perms & rolePerms) != 0) {
   log.debug("checkProperties: {}", true);
   return true;
  }
 }

 log.debug("checkProperties: {}", access);
 return access;
}
}
```

Creating your own Suggestion Analyzer

You can create your own Suggestion Analyzer.

Conditions:

- *The new Suggestion Analyzer class **must implement the "Suggestion"** interface.*
- *The new Suggestion Analyzer class **must be declared under the package "com.openkm.plugin.form.suggestion".***
- *The new Suggestion Analyzer class **must be annotated with "@PluginImplementation"**.*
- *The new Suggestion Analyzer class **must extend "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 plugins 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 can be found 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;
  }
}
```

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

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 in 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.*

getSuggestBoxValuesByFilter(String filterValue)	*List<KeyValue>*	*Returns a list of KeyValue objects.*
getExactKeyValueByClassName(String value)	*KeyValue*	*Returns a KeyValue object.*

Example 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 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 lowercase, 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 controls 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 sometimes means you must migrate older version numbers to the new format, or the 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**.*

Method descriptions

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, increases by default.* *INCREASE_MAJOR: Only shows the option to increase the major version; if not selected, increases by default.* *INCREASE_MAJOR_MINOR: Shows the options to increase major and minor versions; if not selected, 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 3 types of template documents:

- *PDF*
- *OpenOffice*
- *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 with the user's 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 simple variable substitution, loops, and other control structures.

> *In the samples we use only the basic features, but you can learn more in the FreeMarker Manual.*

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

Creating LibreOffice templates

Advantages of using Office and LibreOffice templates:

- *Can produce ODT files.*
- *Less time spent creating the document.*

Disadvantages compared with text templates:

- *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 establish a relation between the template form field and the property group, you must 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.

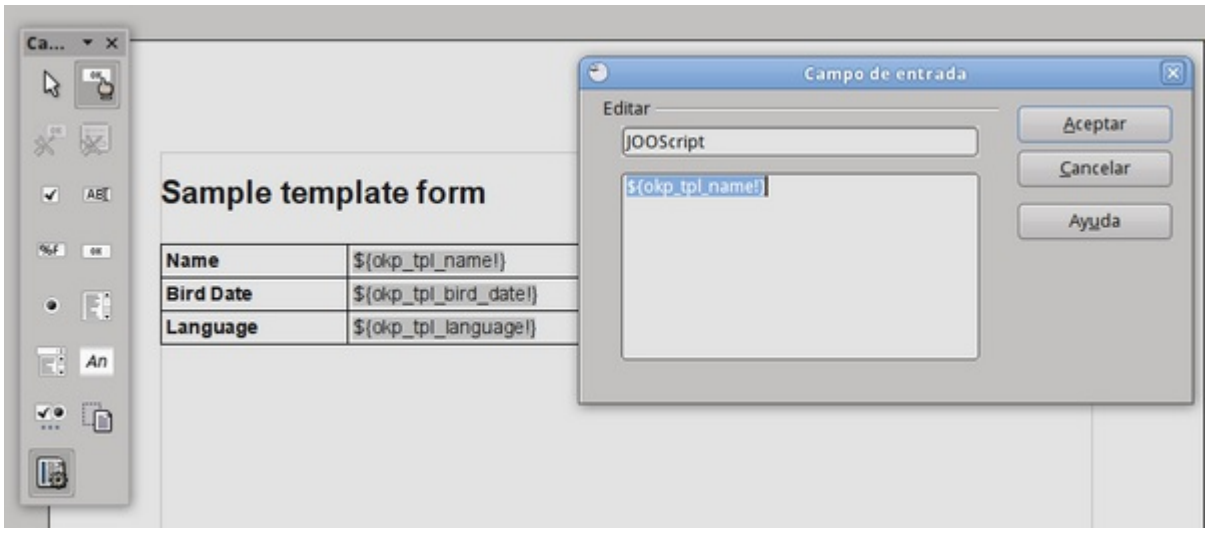
*To create a template in OpenOffice.org, go to **View** > **Toolbars** > **Form Controls**.*

A new dialog box will appear, and you will be able to add form elements to the document. This dialog box contains elements such as labels, text boxes, check boxes, etc.

The field that will be replaced with the value of a passed variable 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 OpenOffice.org Writer from the menu **Insert** / **Fields** / **Other...** (or just press **Ctrl+F2**), then click on the **Functions tab** and select **Input field**. Change the field Reference to **JOOScript**. In the following 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; hovering the mouse over it will reveal the field reference.*



This is the OpenOffice.org document used in this sample tpl.odt.

Read about OpenOffice.org templates at JODReports.

Date format

JODReports uses FreeMarker, so you can take advantage of FreeMarker's formatting flexibility. For example, you can set the date format in this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")}
```

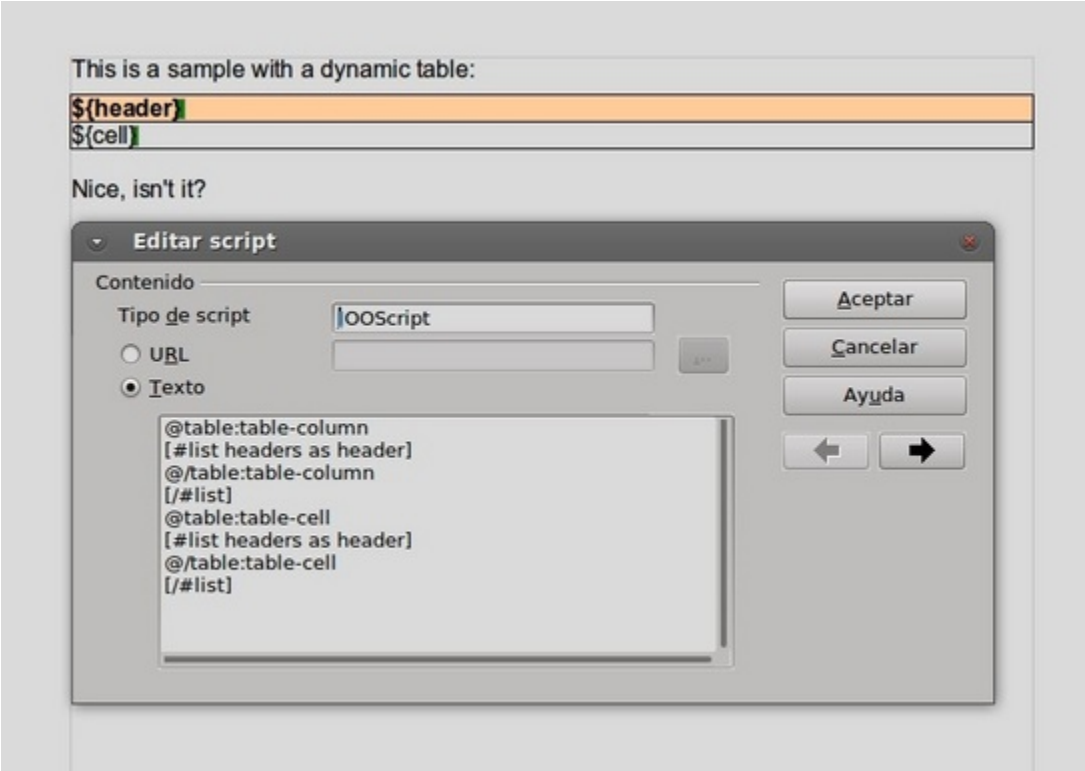
If the date may be null, you need to check for this (in the case of ODT templates, use #if between "[" and "]").

```
[#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 OpenOffice.org 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 the sample

- ***Upload** the **tpl.odt** file to **/okm:templates/documents***
- ***Register the metadata** definition **okg:tpl** at **Administration** > Metadata*
- ***Enable metadata** for **profiles** at **Administration** > Profiles*

- ***Add the metadata*** *group okg:tpl to the tpl.odt file.*

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

Creating a PDF template

Advantages of using PDF templates:

- *Can produce PDF files.*

Disadvantages in comparison 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 form templates with several applications:

- *Acrobat Pro*
- *How-to: Create PDF forms with OpenOffice*
- *PDF Forms Designer*

Every PDF form field name needs to be unique; you can't set the same property value for several form fields.

When converting an OpenOffice document to PDF, repeated form fields are renamed to original_name_2, original_name_3, etc.

If you need to put the same property value in several fields, try Creating LibreOffice templates.

In the sample below, we've used OpenOffice to create a PDF template.

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>
```

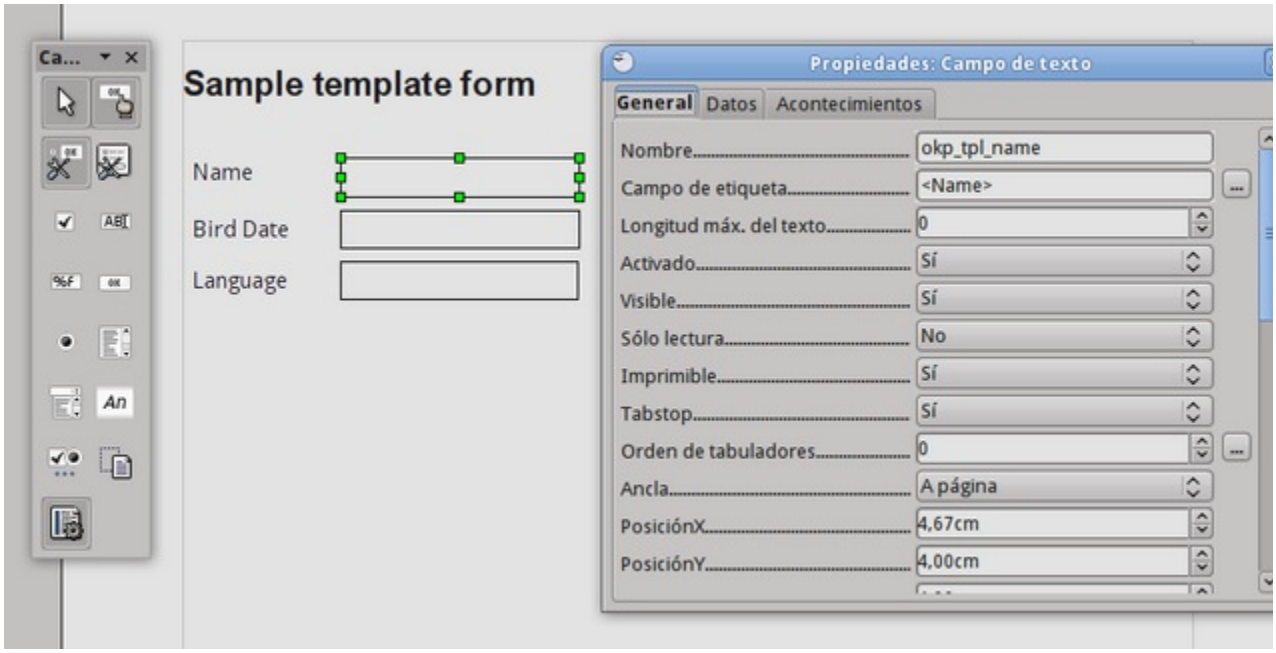
In order to create a relationship between the template form field and the property group, you have to name them using the following naming convention: replace ":" and "." with "_". Here you can see the matching between the two fields:

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

*To create a template in OpenOffice.org go to **View** > **Toolbars** > **Form Controls**.*

A new dialogue will appear, and you will be able to add form elements to the document. This dialogue contains elements such as labels, text boxes, check boxes, etc.



*Once the form has been created, go to **File** > **Export as PDF** to generate a PDF with these form fields. This is the final result:*

Here is the generated PDF file tpl.pdf and here is the ODT source tpl_pdf.odt.

Date format

OpenKM uses FreeMarker internally, so you can take advantage of FreeMarker's formatting features. For example, you can set the date format this way:

```
${okp_tpl_birth_date?string("yyyy-MM-dd")}
```

Learn more about FreeMarker formatting at Built-in Reference.

Configuring sample

- ***Upload** the **tpl.pdf** file to **/okm:templates/documents***
- ***Register the metadata** definition **okg:tpl** in **Administration** > Metadata*
- ***Enable metadata** for **profiles** in **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 establish 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

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 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 provide a default value if one is missing using 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 internally, so you can take advantage of FreeMarker's formatting flexibility. For example, you can set the date format 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 in the package "com.openkm.validator.password".***

Password validator interface:

```
package com.openkm.validator.password;

import com.openkm.validator.ValidatorException;

public interface PasswordValidator {

    /**
     * Validate
     *
     * @param password
     * @throws ValidatorException
     */
    void Validate(String password) throws ValidatorException;

}
```

From 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 for generating reports. To build reports, you have a tool named iReport.

*The **JasperReports Library** is the world's most popular open-source reporting engine. It is entirely written in Java and can use data from any kind of data source and produce pixel-perfect documents that can be viewed, printed, or exported in a variety of document formats including HTML, PDF, Excel, OpenOffice, and Word. More information at http://community.jaspersoft.com/.*

***iReport** is the free, open-source report designer for JasperReports and JasperReports Server. Create very sophisticated layouts containing charts, images, subreports, crosstabs and much more. Access your data through JDBC, TableModels, JavaBeans, XML, Hibernate, CSV, and custom sources. Then publish your reports as PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX, or OpenOffice. More information at http://community.jaspersoft.com/project/ireport-designer.*

In this section you will find an introduction to creating reports with the iReport tool. Consider it a quick introduction, not a complete guide. You should follow some tutorials and read books or papers to build on it.

There are some books and guides available that 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 reports based on SQL.

- *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.*

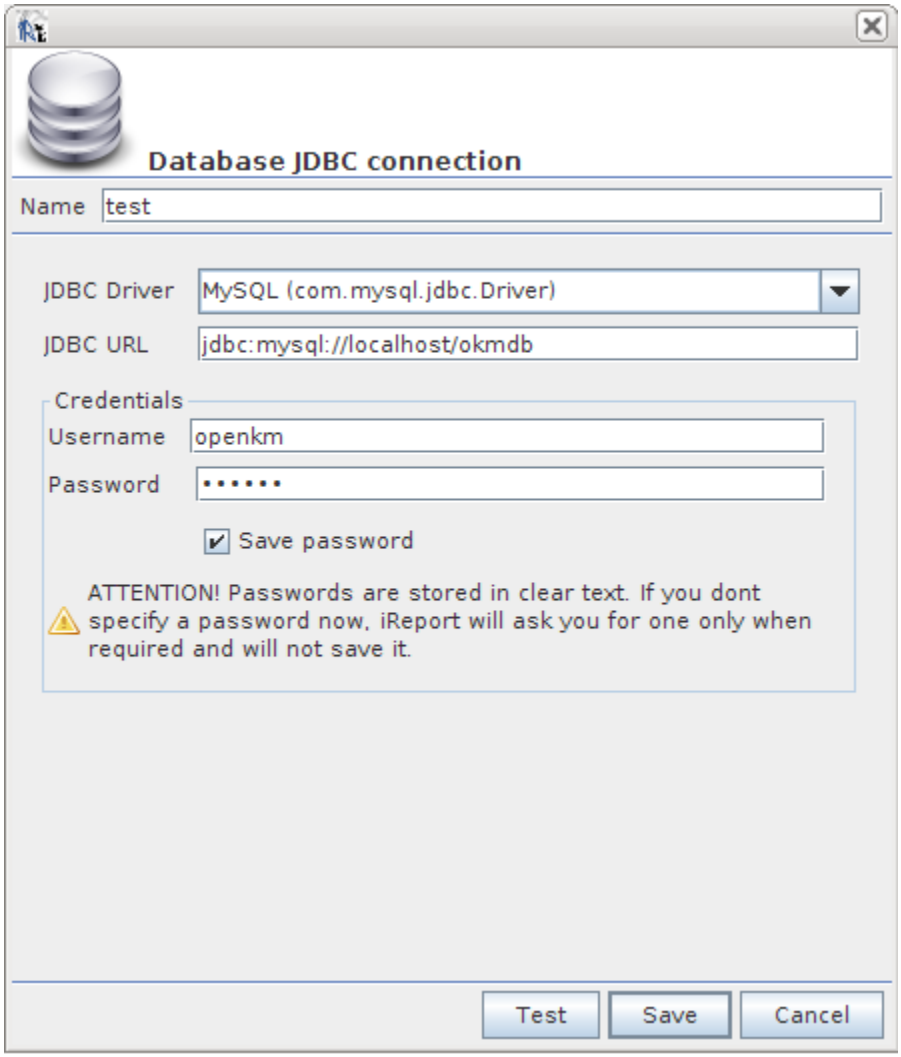
> *OpenKM comes with a specific JasperReports Library engine that supports older versions but not newer ones. See the Application version compatibility table for a correspondence list between OpenKM and JasperReports Library engine versions.*
>
> *You can set a specific output version in the iReport tool:*
>
> - *Click on the **Tools menu > Options**.*
> - *From the **iReport > General tab**, click on the **Compatibility tab**.*
> - *Set the **Jasper version compatibility**.*
> - *Click the **OK button**.*

Create a datasource

- *Click the **Report datasources button**.*

- *Click the **New button**.*
- *Choose the "**Database JDBC connection**" option.*
- *Click the **Next button**.*
- *Set the **name**.*
- *Choose the **JDBC driver**.*
- *Click the **Test button** to check the connection parameters.*
- *Click the **Save button**.*

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

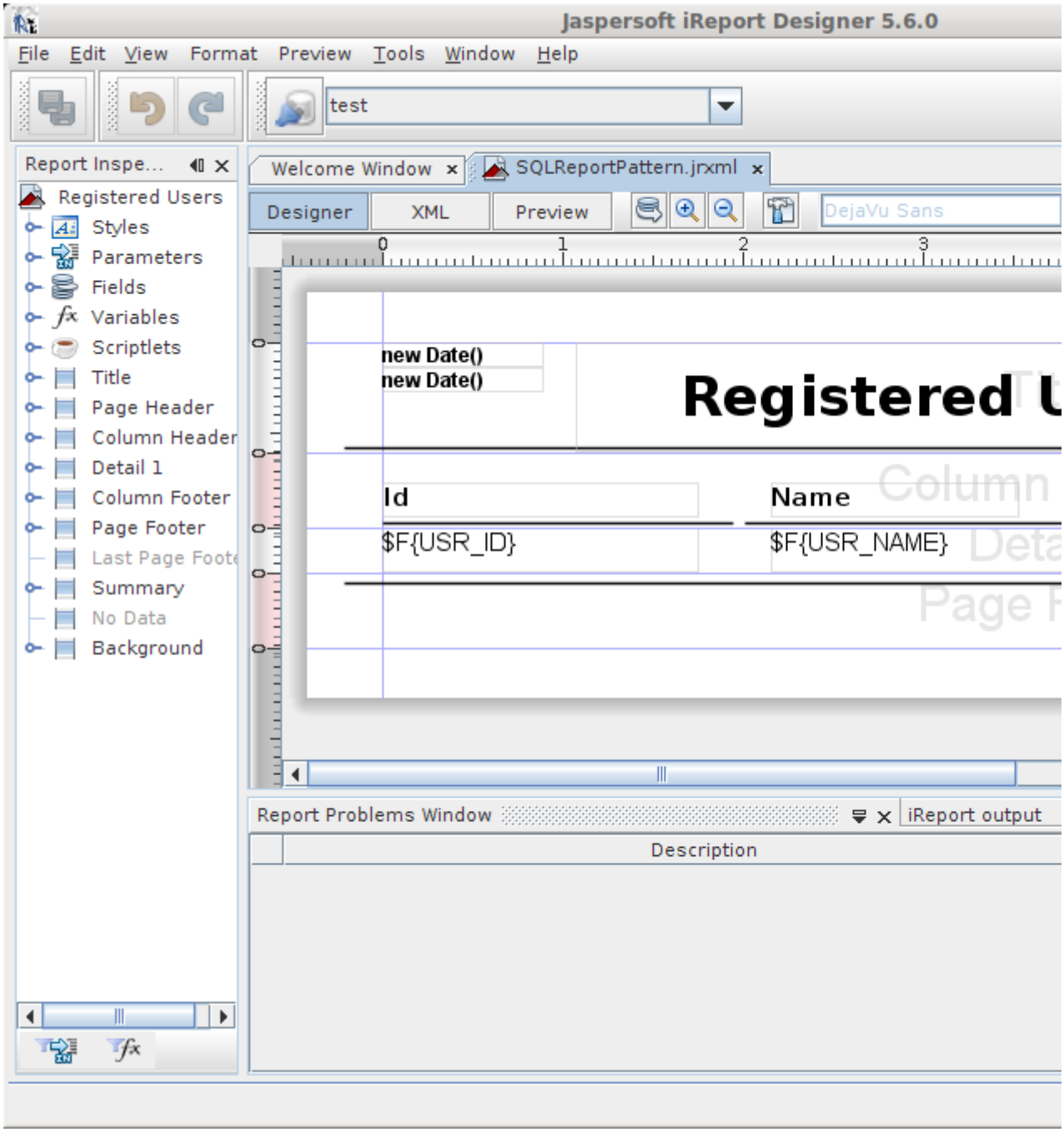


- *Check the **Default** checkbox.*
- *Click the **Close button**.*

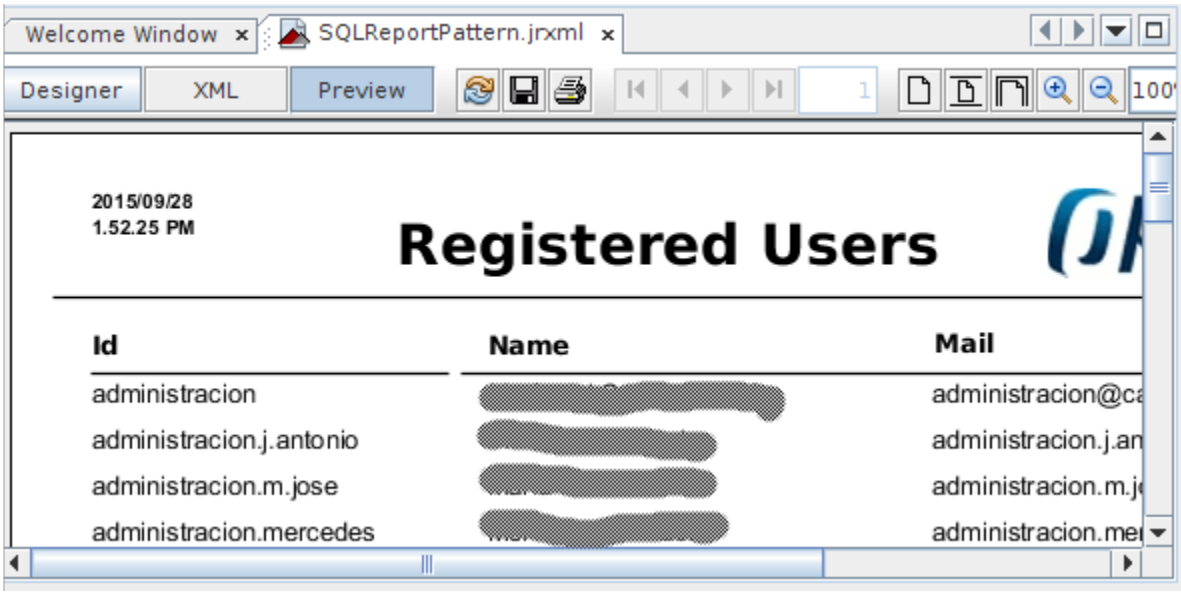
Testing SQL sample report

> *To test the SQL report sample, you must have OpenKM running.*

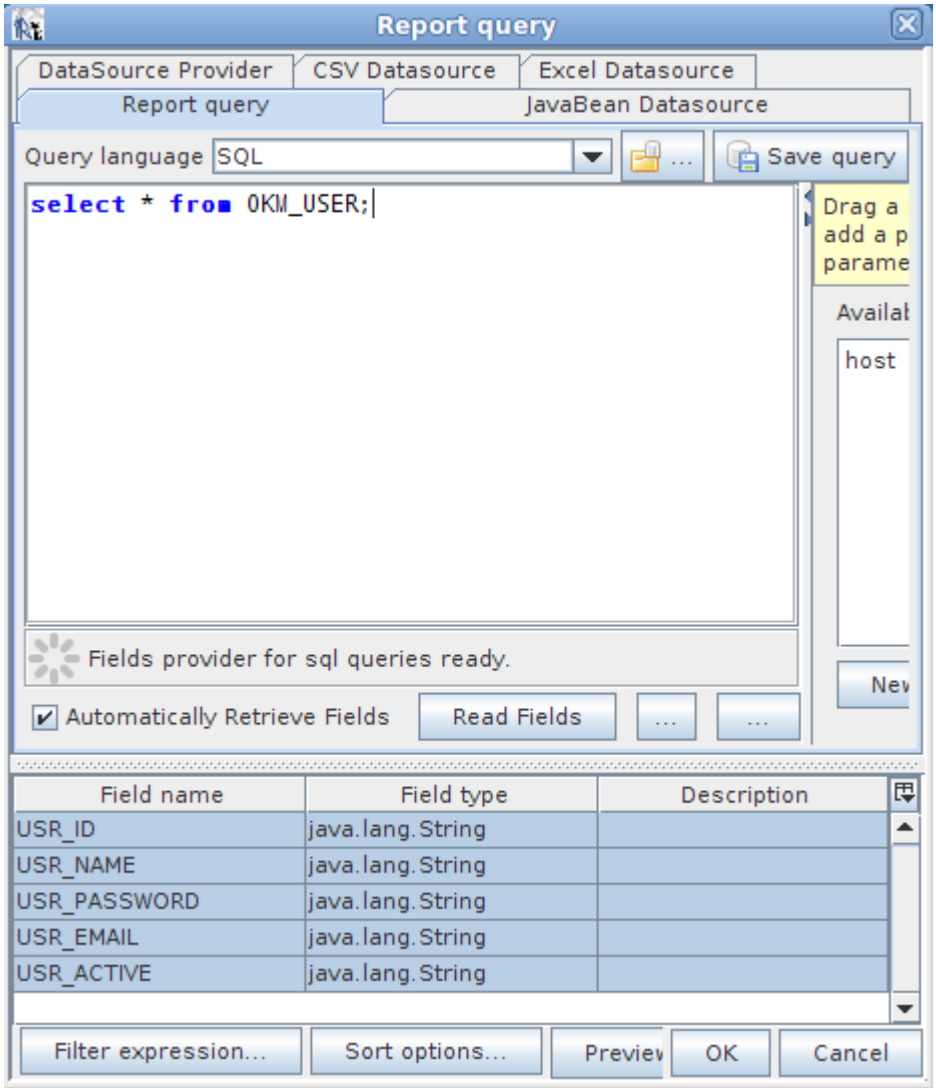
- *Download the [SQLReportPattern.jrxml](#) file.*
- *Open the file.*



- *Click the **Preview tab**.*
- *Set the **host** value (for example http://localhost:8080/OpenKM).*



- *Click the **Designer tab**.*
- *Click 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 what kind of file is 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-aab9-9adb97b
    <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

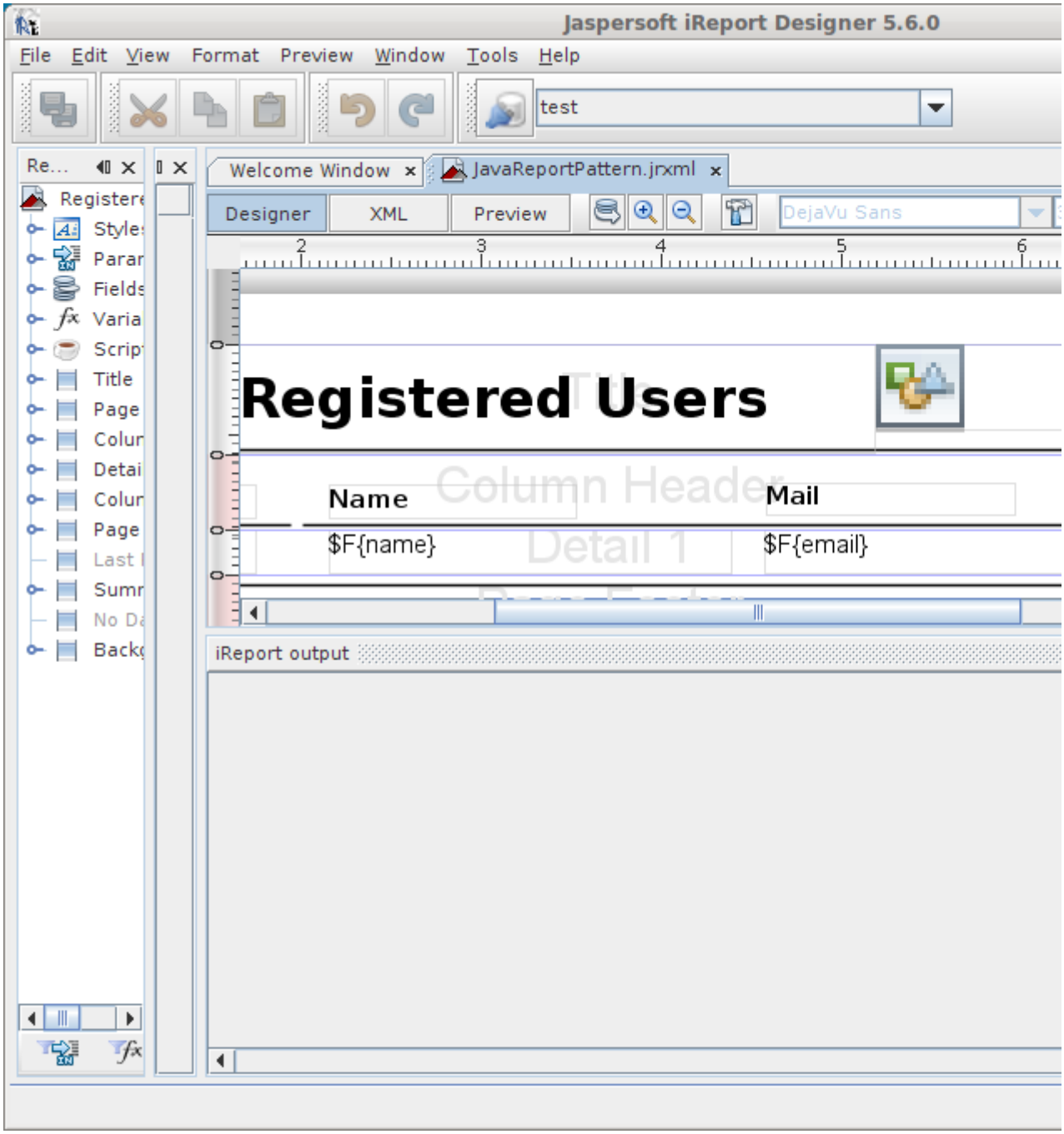
- *Install the **iReport** tool.*
- ***Execute the iReport** tool.*

> *OpenKM comes with a specific JasperReports Library engine that supports older versions, but a newer one is now available. In the Application version compatibility table you have a correspondence list between OpenKM and JasperReports Library engine versions.*
>
> *You can set the specific output version in the iReport tool:*
>
> - *Click on the **Tools menu > Options**.*
> - *From the **iReport > General tab**, click on the **Compatibility tab**.*
> - *Set the **Jasper version compatibility**.*
> - *Click on the **OK button**.*

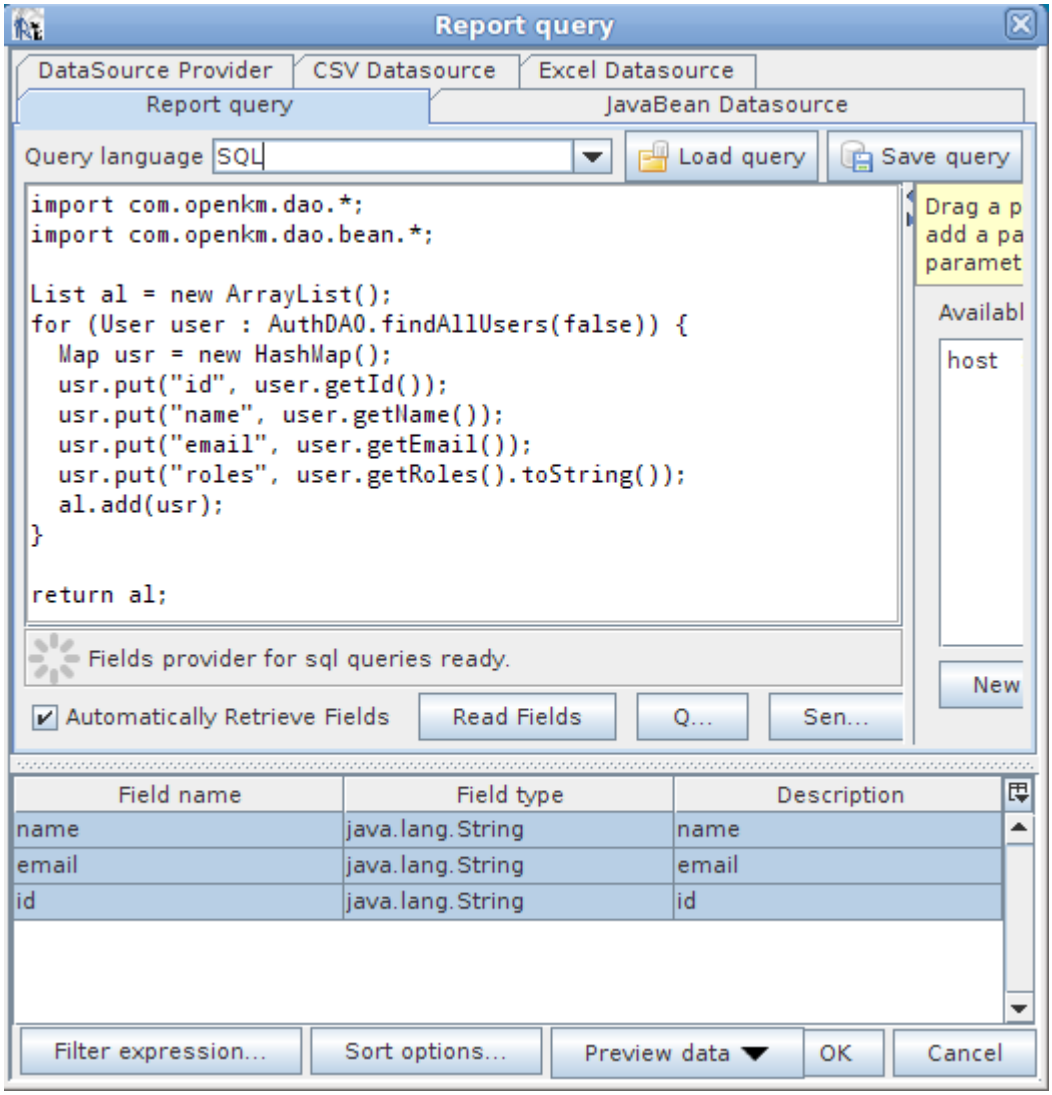
Testing the Java sample report

> ***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 OpenKM running.*

- *Download the JavaReportPattern.jrxml file.*
- *Open the file.*



- *Click on the **Designer tab**.*
- *Click on* *the **report query button**.*



In this case, instead of an SQL query, you can use 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.*

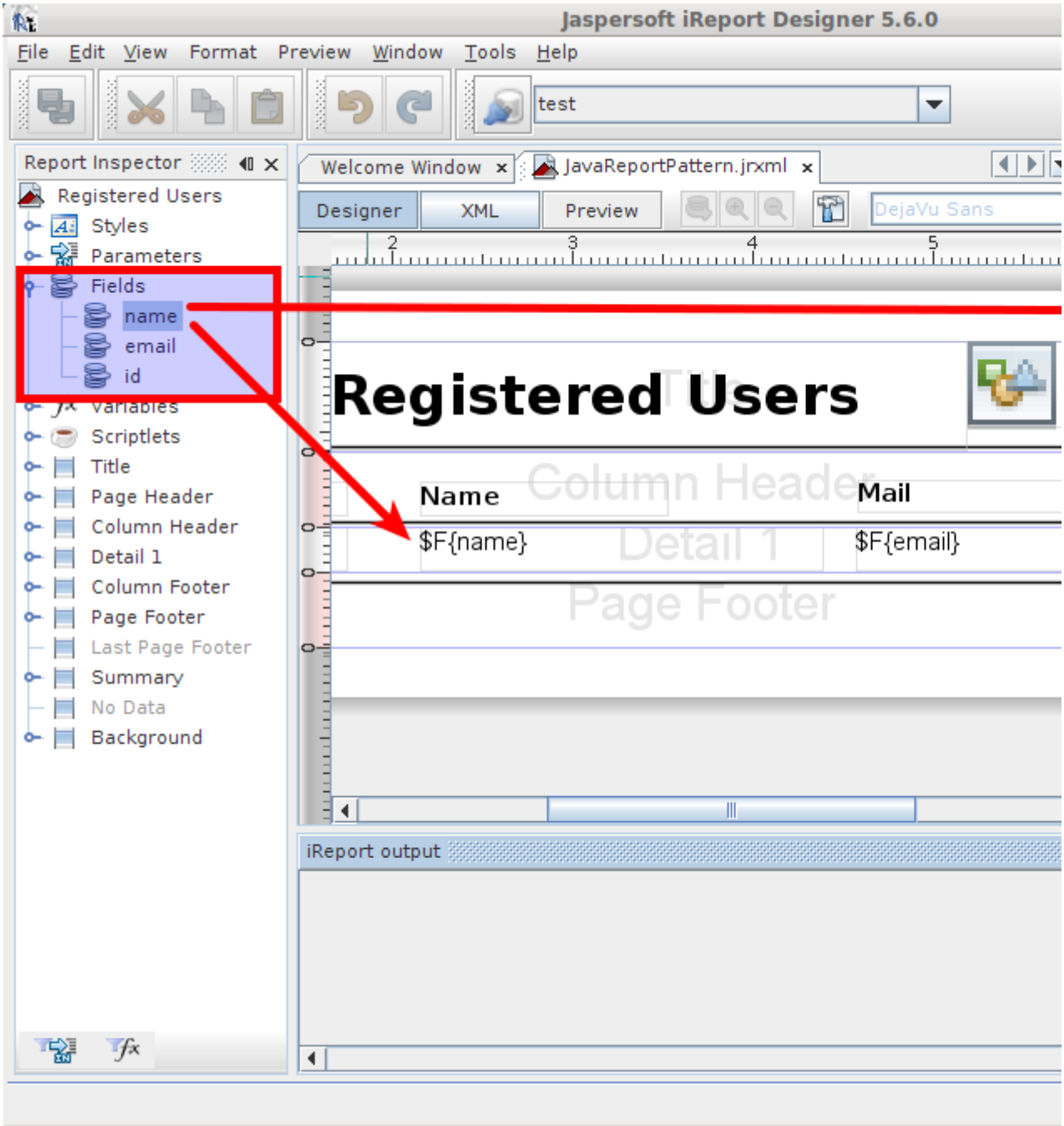
```
import com.openkm.db.service.*;
import com.openkm.bean.*;
import com.openkm.util.ContextWrapper;
import org.springframework.web.context.WebApplicationContext;
```

```
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 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.


Jaspersoft iReport Designer 5.6.0
File Edit View Format Preview Window Tools Help
test
Report Inspector
Registered Users
Styles
Parameters
Fields
name
email
id
Variables
Scriptlets
Title
Page Header
Column Header
Detail 1
Column Footer
Page Footer
Last Page Footer
Summary
No Data
Background
Welcome Window
JavaReportPattern.jrxml
Designer
XML
Preview
DejaVu Sans
Registered Users
Name
Mail
$F{name}
$F{email}
iReport output

Reports with parameters

When you execute a report from OpenKM the view might show a form requiring the user to enter values for filtering purposes.

The most common case is filtering by a range of dates.

> *Reports with form filtering end with the extension "**.rep**". It is actually a ZIP file renamed to "**.rep**".*
>
> *These reports contain two files:*
>
> - *JRXML file*
> - *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 for filtering from date.
to_date	**Input**	Field used for filtering to date.

The params.xml file:

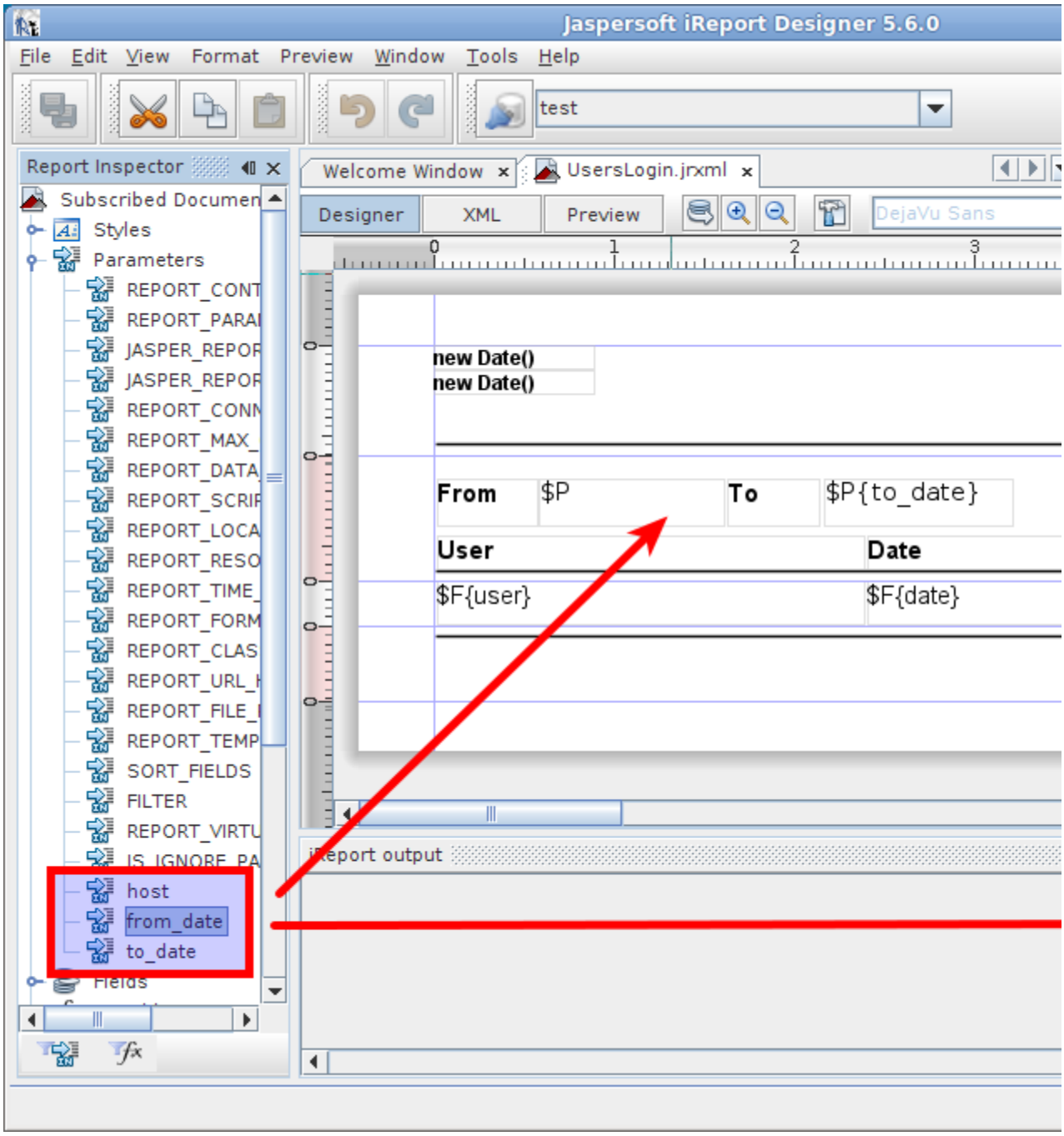
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <input label="From" name="from_date" type="date">
    <validator type="req"/>
  </input>
  <input label="To" name="to_date" type="date">
    <validator type="req"/>
  </input>
</report-parameters>
```

Using fields in report

Two parameters are created in the report:

Parameter	Class	Description

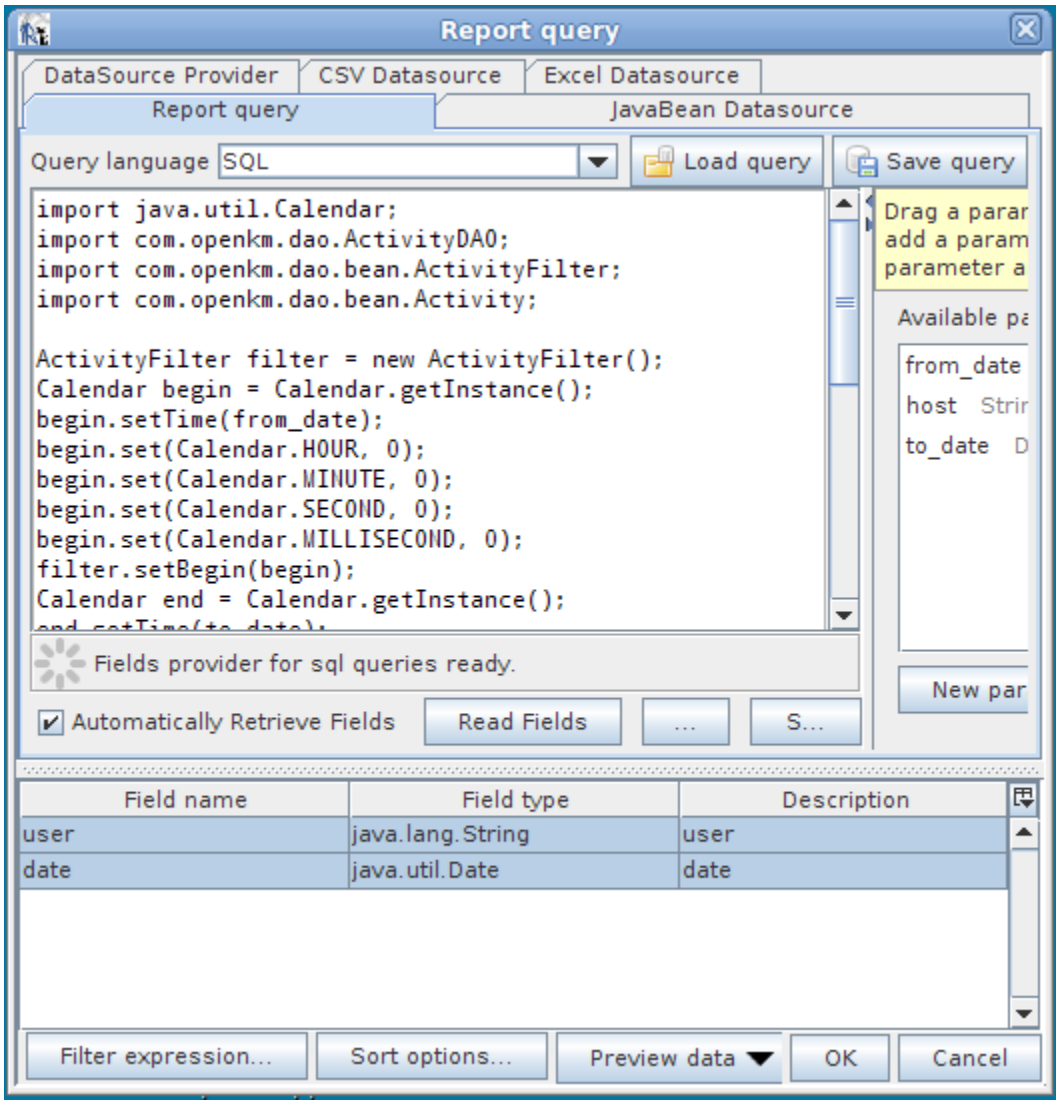
from_date	***java.util.Date***	*From date.*
to_date	***java.util.Date***	*To date.*



The parameters are also used in SQL or Java reports.

Java sample

- *Click on* ***report query button****.*



The ***from_date*** *and* ***to_date*** *variables are injected into code execution and can be used directly as in the sample below.*

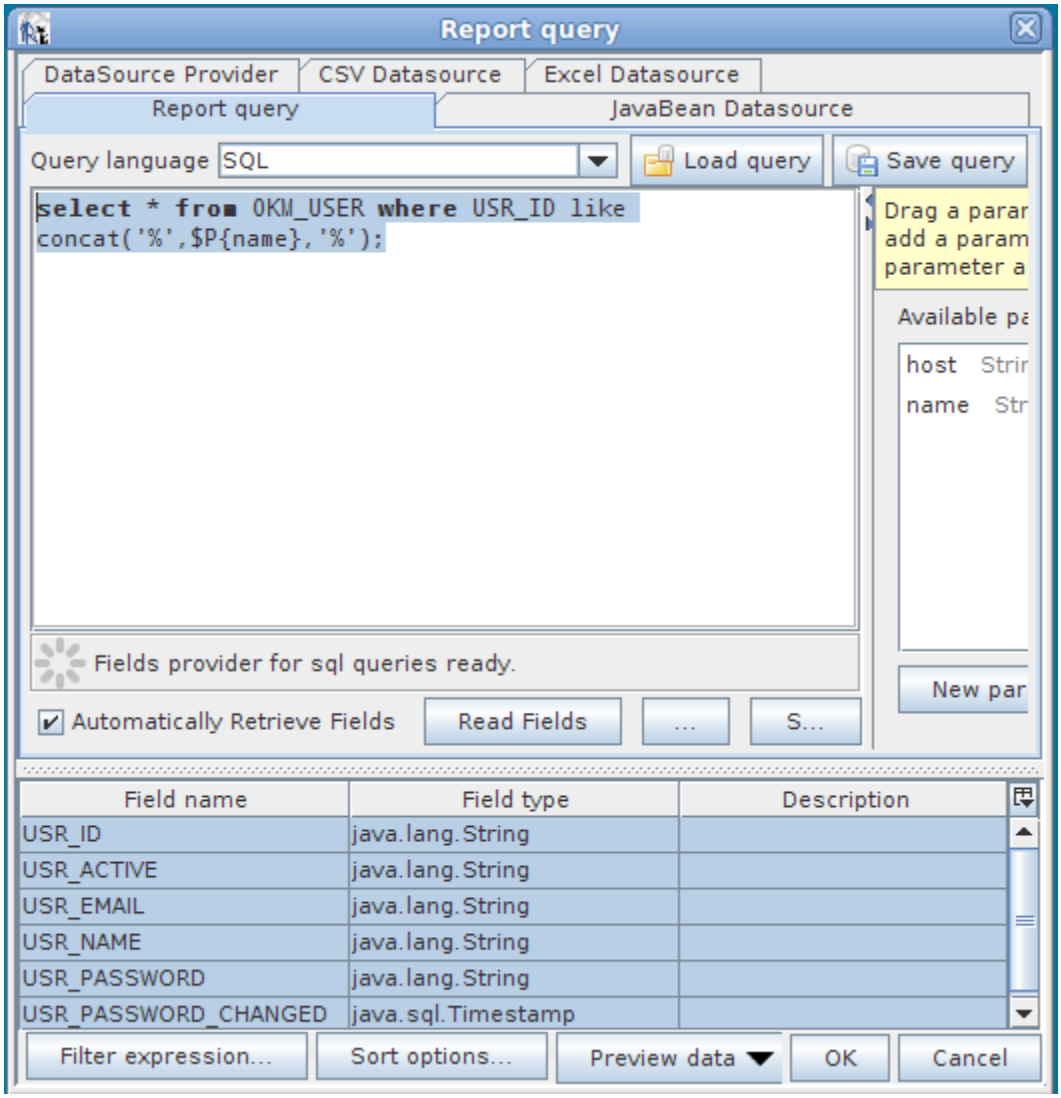
```
import java.util.Calendar;
import com.openkm.dao.ActivityDAO;
import com.openkm.dao.bean.ActivityFilter;
import com.openkm.dao.bean.Activity;

ActivityFilter filter = new ActivityFilter();
Calendar begin = Calendar.getInstance();
begin.setTime(from_date);
begin.set(Calendar.HOUR, 0);
begin.set(Calendar.MINUTE, 0);
begin.set(Calendar.SECOND, 0);
begin.set(Calendar.MILLISECOND, 0);
filter.setBegin(begin);
Calendar end = Calendar.getInstance();
end.setTime(to_date);
end.set(Calendar.HOUR, 0);
```

```
end.set(Calendar.MINUTE, 0);
end.set(Calendar.SECOND, 0);
end.set(Calendar.MILLISECOND, 0);
filter.setEnd(end);
filter.setAction("LOGIN");
List al = new ArrayList();

for (Iterator it = ActivityDAO.findByFilter(filter).iterator(); it.hasNext();){
    Activity act = (Activity) it.next();
    Map actMap = new HashMap();
    actMap.put("user", act.getUser());
    actMap.put("date", act.getDate().getTime());
    al.add(actMap);
}

return al
```

SQL sample

The sample below is not based on the UsersLogin.rep report.

Click on ***report query button****.*



*In the sample there is a parameter named "**name**" of class "**java.lang.String**" used for filtering the users list.*

```
select * from OKM_USER where USR_ID like concat('%',$P{name},'%')
```

The engine replaces the "$P{name}" tag with the 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%')
```

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 few DTD definitions; read Application version compatibility table to see which are supported by your application.*

Basic XML skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>

</report-parameters>
```

- *The **DOCTYPE** is a formal definition of some DTD. In the previous example, the **report-parameters-2.1.dtd** is used.*
- *The XML begins with **<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 can be found 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.*

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

Params fields

These are the available field types:

Field type	*Description*
Checkbox	*Params Checkbox field*
Input	*Params Input field*
Select	*Params Select field*
Separator	*Params Separator field*
Suggestbox	*Params Suggestbox field*
Text	*Params Text field*
TextArea	*Params TextArea field*

Some field types can have child, validator, or option elements. This hierarchy is permitted:

- *Params form*
  - *Checkbox*
    - *Validator*
  - *Input*
    - *Validator*
  - *Select*
    - *Option*
    - *Validator*
  - *Separator*
  - *Suggestbox*
    - *Validator*
  - *Text*
  - *TextArea*

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

- *Validator*

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

Params Validator element

The goal of the Validator element is to ensure that the user has provided 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
req	*Required value.*
alpha	*Alphabetic value.*
alphanum	*Alphanumeric value.*
dec	*Decimal value.*
num	*Numeric value.*
email	*Email value.*
url	*URL value.*
maxlen	*Set the maximum length allowed.*
minlen	*Set the 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:

- *Params Input field.*
- *Params Suggestbox field.*
- *Params Checkbox field.*
- *Params TextArea field.*
- *Params Select field.*

Basic example

- *Value required: "**req**".*
- *Value must be a 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"/>
    <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.

Params Input field

In most cases, it is used to enter free-form text, although it can be used to store 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 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>
```

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 chars in input to begin to filter results.*	*true*

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

dialogTitle	*The title of the HTML dialog.*	*true*

Child elements:

- *Validator. For more information, see [Params Validator element](#).*

** There are three options to get suggested box values (using **one of them is mandatory**):*

- *table + filterQuery + valueQuery*
- *filterQuery + valueQuery*
- *className*

No other combination of them is allowed.

Based on the table, filterQuery, and valueQuery

Suggestbox values are defined in metadata tables.

- *The application uses two tables named OKM_DB_METADATA_TYPE and OKM_DB_METADATA_VALUE to help store option values used in selects. That is 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 may differ depending on your database; 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 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 retrieve a list from external applications or for other purposes, your class can be built to get the option values from those systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
   <suggestbox label="country" name="okp:consulting.suggestbox"
                width="200px" dialogTitle="Choose country" filterMinLen="3"
                className="com.openkm.plugin.form.values.SuggestBoxUserList" />
</report-parameters>
```

For more information, read Creating your own Suggestbox plugin.

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 %, 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	*A 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*

Child elements:

- *Validator. For more information, see the [Params Validator element](#).*

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                   "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <textarea label="textarea label" name="textarea" width="200px" />
</report-parameters>
```

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

Params Select field

Attribute	*Description*	*Required*
label	*The text is shown as a label in the user interface.*	*true*
name	*Unique field identifier.* *Two fields can't have the same name in the 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 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**" from 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 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 report-parameters PUBLIC "-//OpenKM//DTD Report Parameters 2.1//EN"
                                "http://www.openkm.com/dtd/report-parameters-2.1.dtd">
<report-parameters>
  <select label="country" name="country" type="simple"
        optionsQuery="select CT_ID, CT_NAME from COUNTRY order by CT_NAME" />
</report-parameters>
```

The table used in this sample is defined as follows:

```
create table COUNTRY (
    CT_ID varchar(2),
    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 retrieve 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>
```

```
</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 percentages, for example:* • *width="50%" will work* • *width="100px" will be ignored by 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	*A 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 has an API exposed via web services with:

- *RESTful*.

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

RESTful

OpenKM has a complete API exposed via REST. You can call any API method from any programming language, like Java, PHP, or Python. This feature makes it possible to create a custom client or integrate with third-party applications like a CRM or a CMS.

> *These URLs are protected by BASIC authentication, so you must provide a 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 using our SDKs to access OpenKM via the REST API instead of building your own client; more information:*
>
> - *The SDK for Java.*
> - *The SDK for .NET.*

*In almost all samples, you'll see parameters named "**docId**," "**fldId**" or "**nodeId**." The value of this parameter can be a valid node **UUID (from version 7.x, we have deprecated almost all methods where the path can be used)** .*

> *Example of nodeId:*
>
> - *Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";*
> - *Using path -> "**/okm:root/logo.png**"*

> *Almost all samples use the curl command; more information is available 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 this 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",
    "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
```

Alternatively, from an HTML form:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document Upload</title>
</head>
<body>
    <form method="POST" enctype="multipart/form-data"
          action="http://localhost:8080/openkm/rest/documents/parents/fe239ae2-4d25-4d87-88b2-924d03b87faa">
      Select file: <input type="file" name="content" size="45"/><br/>
      <input type="submit" value="Upload" />
    </form>
</body>
</html>
```

And now download it:

```
$ curl -u okmAdmin:admin \
  http://localhost:8080/openkm/rest/documents/dab830ea-2fa8-439f-837e-de15409d88b9/content
```

Search samples

Search with parameters:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/search?keyword=test&name=doc*&mimeType=application/pdf
```

Search with metadata:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/search?keyword=test&property='okp:consulting.name=alfa'
```

Security samples

Show granted users:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/users
```

Show granted roles:

```
$ curl -u okmAdmin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/roles
```

Add a grant:

```
$ curl -v -u okmAdmin:admin -X POST -H "Content-Type: application/x-www-form-urlencoded" \
```

```
-d userId=sochoa -d permissions=15 -d recursive=false \

http://localhost:8080/openkm/rest/auth/nodes/fe239ae2-4d25-4d87-88b2-924d03b87faa/permissions/us
```

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

Using the conversion feature of the OpenKM API, you can convert, for example, a Microsoft Word document to PDF, among other conversion types available.

Convert a DOC file to PDF:

```
$ curl -u okmAdmin:admin -F content=@sample.doc -o sample.pdf \
  http://localhost:8080/openkm/rest/conversion/doc/pdf
```

Alternatively, from an HTML form:

```
<html>
  <body>
    <form method="POST" enctype="multipart/form-data"
          action="http://localhost:8080/openkm/services/rest/conversion/doc2pdf">
      Select file: <input type="file" name="content" size="45"/><br/>
      <input type="submit" value="Convert" />
    </form>
  </body>
</html>
```

Convert a DOC file to TXT:

```
$ curl -u okmAdmin:admin -F content=@sample.doc http://localhost:8080/openkm/rest/conversion/doc/txt
```

Convert an image file to TXT:

```
$ curl -u okmAdmin:admin -F content=@sample.tif http://localhost:8080/rest/rest/conversion/img/txt
```

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

Swagger

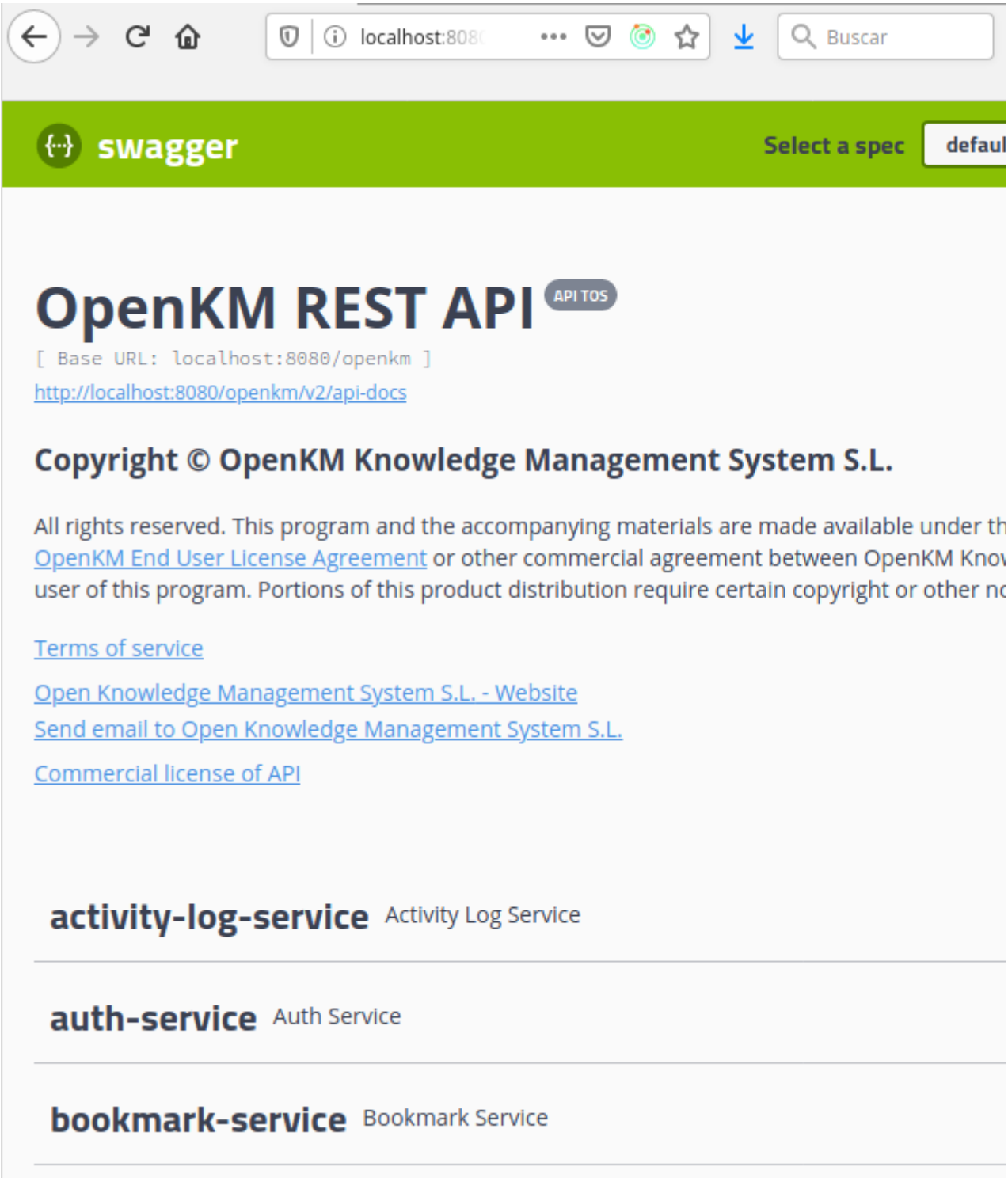
***Swagger** is an open-source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services. While most users identify Swagger by the Swagger UI tool, the Swagger toolset includes support for automated documentation, code generation, and test case generation.*

How to access Swagger

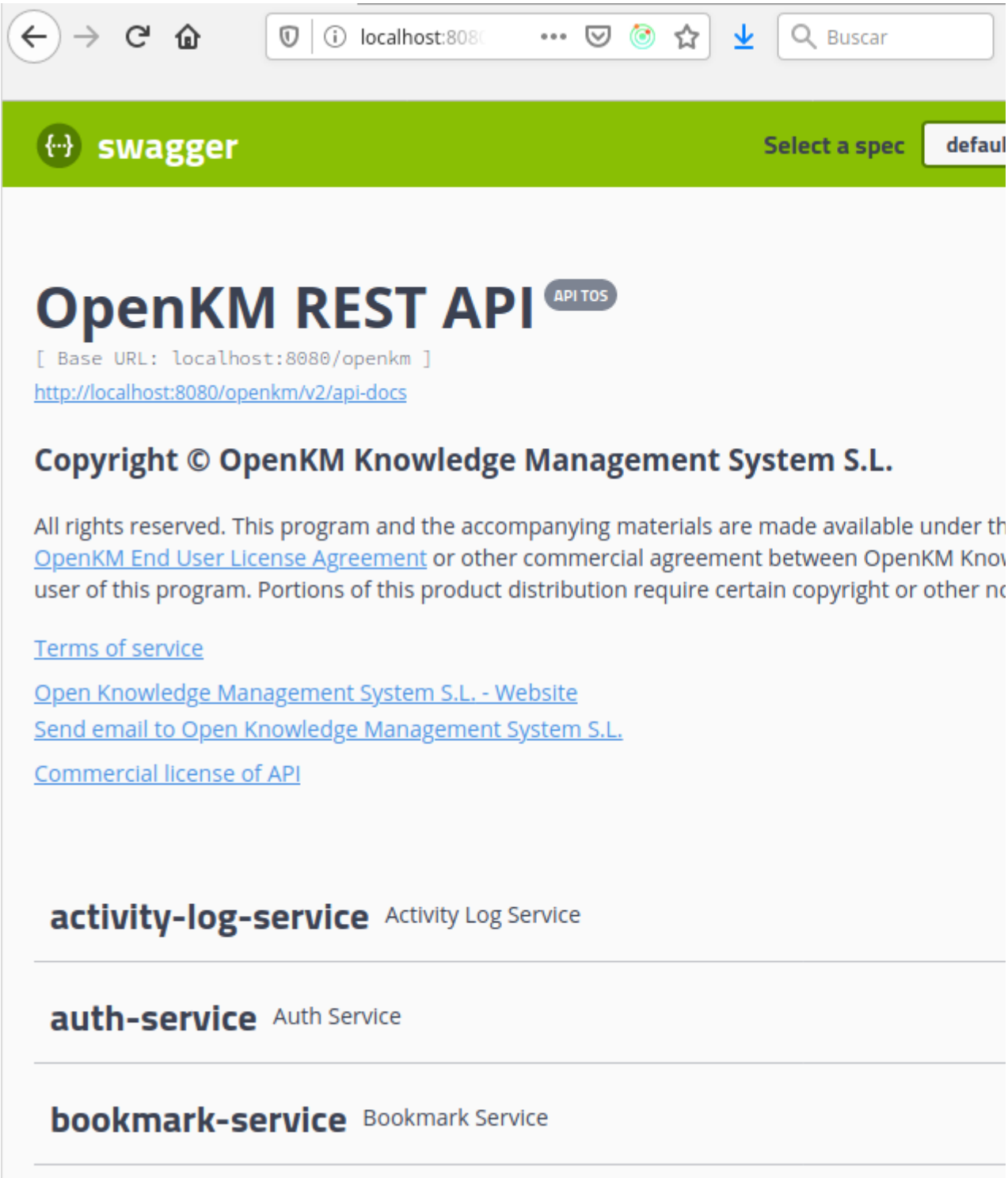
*In the **sample below, we use our OpenKM demo URL**. You **should use your OpenKM URL**.*

To access the Swagger documentation, click the URL http://localhost:8080/openkm/swagger-ui.html it will display a screen like the one shown below asking for authorized user credentials:

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



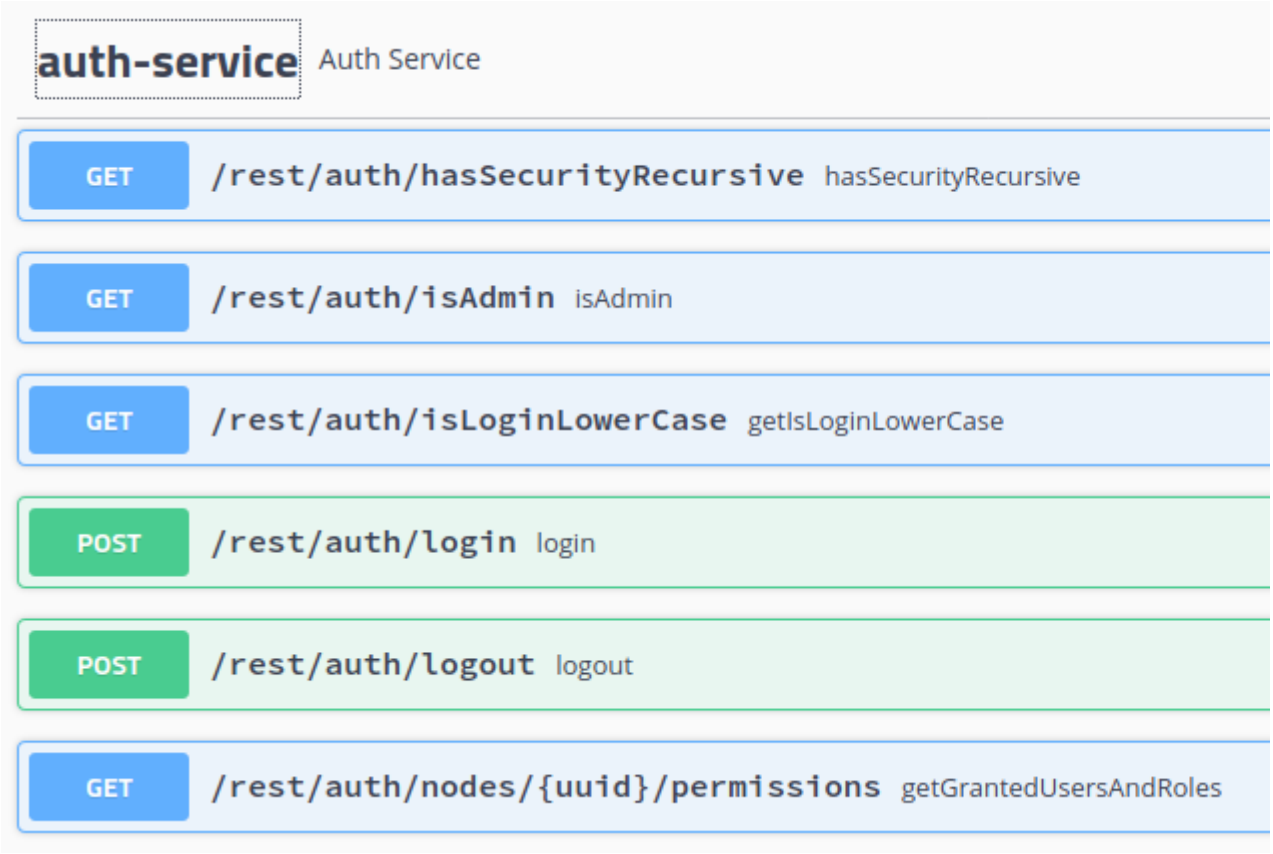
Then the Swagger tool will be shown:



Playing with Swagger

*Let's do a **quick test** of the tool **checking the getRoles** method **from the auth-service**.*

- ***Open the auth-service***



- ***Open the "roles"*** *method and* ***click on the button "Try it out"*** *:*

GET /rest/auth/roles getRoles

Parameters

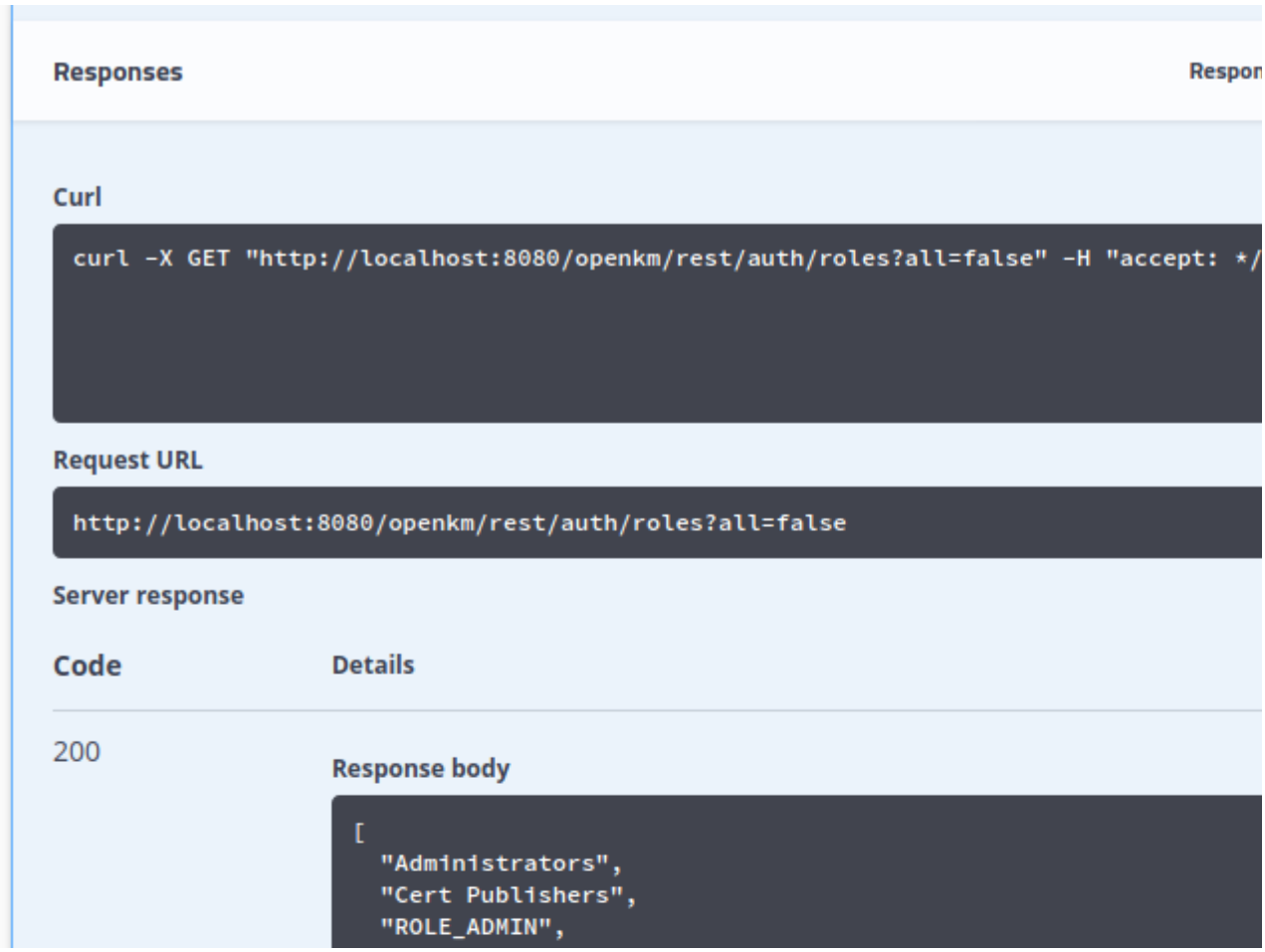
Name	Description
all boolean (query)	all *Default value* : false

Responses Respo

Code	Description
200	OK Example Value | Model ["string"]
401	Unauthorized
403	Forbidden
404	Not Found

- ***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

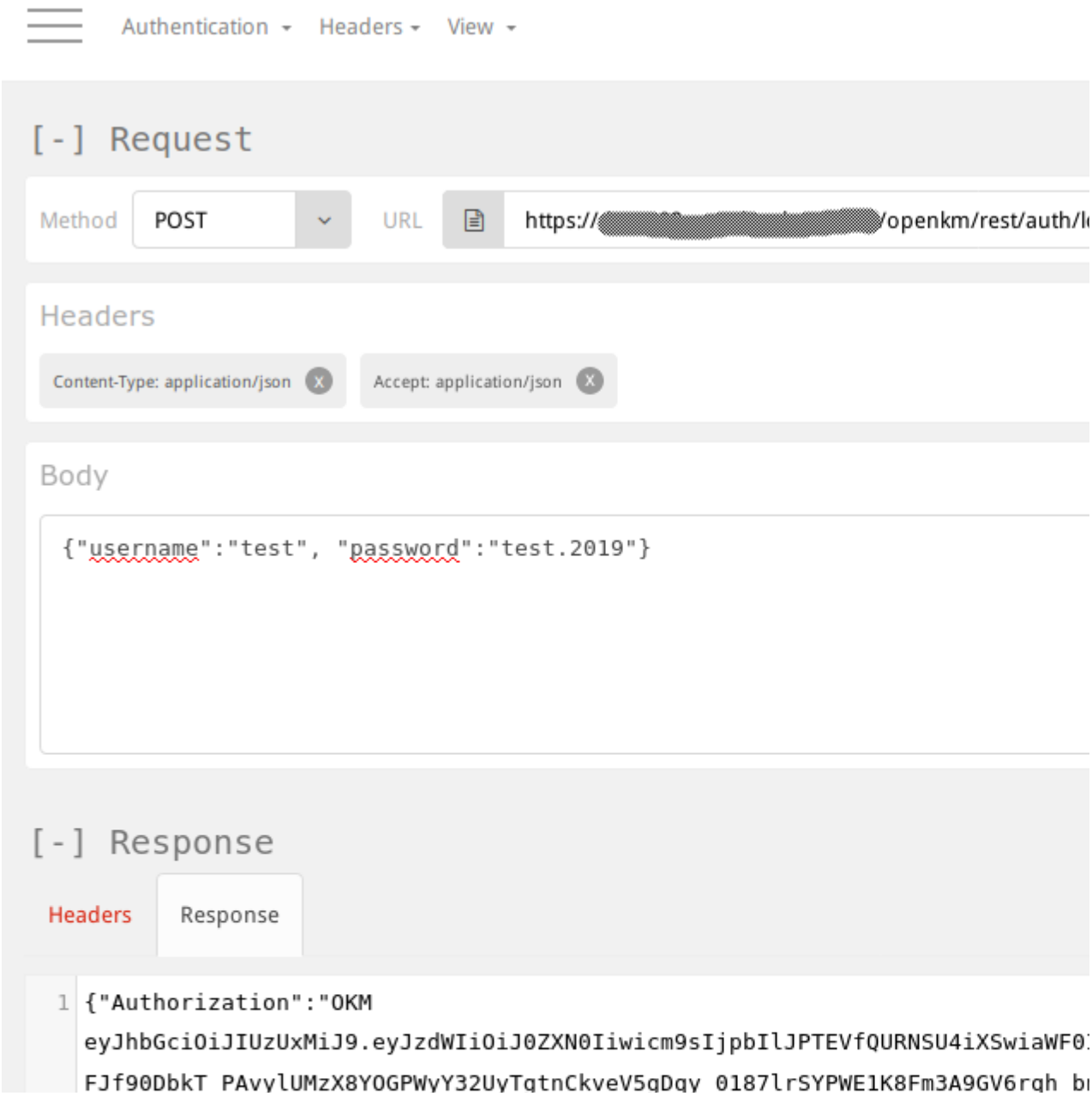
Code	Description
200	*OK*
	Example Value | Model
	`[ "string" ]`
401	*Unauthorized*
403	*Forbidden*
404	*Not Found*

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.*



[-] Curl

Command

```
curl -X POST -k -H 'Content-Type: application/json' -H 'Accept: application/json' -i '
/login' --data '{"username":"test", "password":"test.2019"}'
```

Home Fork me Report an issue

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.*

Authentication Headers View

[-] Request

Method GET URL https://[redacted]/openkm/rest/auth/r

Headers

Content-Type: application/json Accept: application/json Authorization: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0Iiwi

Body

Request Body

[-] Response

Headers Response Preview

```
1 ["ROLE_ADMIN","ROLE_GENERAL","ROLE_LECTURA","ROLE_MANAGER","ROLE_SECURITY_
ER"]
```

[-] Curl

Command

```
curl -X GET -k -H 'Content-Type: application/json' -H 'Accept: application/json' -H 'A
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJ0ZXN0Iiwicm9sIjpbIlJPTEVfQURNSU4iXSwiaWF0IjoxNTY3NjY4M
X8YOGPWyY32UyTgtnCkveV5gDqy_0187lrSYPWE1K8Fm3A9GV6rqh_bnvI4Us2g' -i 'https://[redacted]
```

PHP Samples

Download sdk4php

The sdk4php sample below includes the most common methods used by users when integrating their 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, ask for them in the OpenKM community forum at https://forum.openkm.com*

Professional	PHP version	Status	Download
Version 7.0.0 and above	7.x	Active	sdk4php-3.0.3.zip

Preview sample

The sample below describes how to integrate the OpenKM preview feature in your PHP application.

Professional	PHP version	Status	Download
Version 7.0.0 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 with 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 "**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 /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($uuid)	***string***	*Retrieves the document content - binary data - of the actual document version.*
Parameters: ***$uuid*** *string type is the uuid of the Document*		

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
```

```
    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        $method = 1;
        $this->ws->login(self::USER, self::PASSWORD);
        $content = $this->ws->document->getContent('b02877cc-bc71-4be9-9bfb-2e50b1732fd3');
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_download.png', 'w+');
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                ob_clean();
                $document = $this->ws->document->getDocumentProperties('b02877cc-bc71-4be9-9bfb-2e50b1732fd3');
                $name = PathUtils::getName($document->getPath());
                header("Cache-Control:  maxage=1");
                header("Pragma: public");
                header("Content-Type: " . $document->getMimeType());
                header('Content-Disposition: attachment; filename="' . $name . '"');
                header("Content-Length: " . strlen($content));
                echo $content;
                break;
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getDocumentChildren

Description:

Method	*Return values*	*Description*
getDocumentChildren($uuid)	***array***	*Returns a list of all documents whose parent is fldId.*
Parameters: ***$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, $dstId)	***void***	*Move a document into a folder or record.*
Parameters: ***$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 parameter newName value 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 "**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();
?>
```

extendedDocumentCopy

Description:

Method	*Return values*	*Description*
extendedDocumentCopy($uuid, $dstId, $newName, $categories, $keywords, $propertyGroups, $notes, $security)	***void***	*Copies a document 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*

$newName *string type is the newName*

$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 categories 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);
            $document = $this->ws->document->extendedDocumentCopy('4ee59dea-4641-4336-a2c4-af384d9040ad',
            var_dump($document);
            echo 'extendedDocumentCopy';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
```

```
}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

checkout

Description:

Method	*Return values*	*Description*
checkout($uuid)	***void***	*Marks the document for editing.*

Parameters:

$uuid *string type is the uuid or path of the Document*

Only one user can modify a document at the same time.

Before starting editing, you must perform a checkout action that locks the editing process for other users and allows it only to the user who executed the action.

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->checkout('eb0e6511-3415-47ca-bcde-393d97944c85');
            // At this point the document is locked for other users except for the user who executed the action
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
```

```
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

cancelCheckout

Description:

Method	*Return values*	*Description*
cancelCheckout($docId)	***void***	*Cancels document editing.*

Parameters:

$docId *string type is the uuid or path of the Document*

This action can only be done by the user who previously executed the checkout action.

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);
            // At this point the document is locked for other users except for the user who executed the action
            $this->ws->document->cancelCheckout('eb0e6511-3415-47ca-bcde-393d97944c85');
            // At this point other users are allowed to execute a checkout and modify the document
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
```

```
}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

forceCancelCheckout

Description:

Method	*Return values*	*Description*
forceCancelCheckout($uuid)	***void***	*Cancels document editing.*

Parameters:

$uuid *string type is the uuid or path of the Document*

This method allows you to cancel editing on any document.

It is not mandatory that this action is executed by the same user who previously executed the checkout action.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
<?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);
            // At this point the document is locked for other users except for the user who executed the action
            $this->ws->document->forceCancelCheckout('eb0e6511-3415-47ca-bcde-393d97944c85');
            // At this point other users are allowed to execute a checkout and modify the document
```

```
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

isCheckedOut

Description:

Method	*Return values*	*Description*
isCheckedOut($uuid)	***bool***	*Returns a boolean that indicates whether the document is being edited or not.*

Parameters:

$uuid *string type is the uuid or path 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 {
            $this->ws->login(self::USER, self::PASSWORD);
            var_dump($this->ws->document->isCheckedOut('eb0e6511-3415-47ca-bcde-393d97944c85'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}
```

```
$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

checkin

Description:

Method	*Return values*	*Description*
checkin($uuid, $content, $comment,$increment)	***Version***	*Updates a document with a new version and returns an object with new Version values.*

Parameters:

$uuid *string type is the uuid or path of the Document*

$content *string type; it is recommended to use file_get_contents â€" reads the entire file into a string.*

$comment *string type is the comment for the new version of the document*

$increment *int type*

The value of the increment variable must be 1 or greater.

The valid values of the increment variable depend on the VersionNumberAdapter you have enabled.

Only the user who started the editing (checkout) is allowed to update 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);
            $this->ws->document->checkout('eb0e6511-3415-47ca-bcde-393d97944c85');
            if ($this->ws->document->isCheckedOut('eb0e6511-3415-47ca-bcde-393d97944c85')) {
                $fileName = dirname(__FILE__) . '/files/logo.png';
                $version = $this->ws->document->checkin('eb0e6511-3415-47ca-bcde-393d97944c85', file_get_contents
                var_dump($version);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getVersionHistory

Folder sample

createFolder

Description:

Method	*Return values*	*Description*
createFolder($uuid, $name)	***Folder***	*Creates a new folder and returns a Folder object as a result.*
Parameters: ***$uuid*** *string type is the destination uuid of the Folder or Record*		

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
```

```
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->createFolder('f93ac8df-e83b-487f-9a14-0af9e910a818', 'test');
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getFolderProperties

Description:

Method	*Return values*	*Description*
getFolderProperties($uuid)	***Folder***	*Returns the folder properties.*

Parameters:

$uuid *string type is the uuid of the Folder*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    // const HOST = "http://localhost:8080/openkm/";
    const HOST = "http://192.168.1.11:8888/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
```

```
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->getFolderProperties("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5");
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

deleteFolder

Description:

Method	*Return values*	*Description*
deleteFolder($uuid)	***void***	*Deletes a folder.*
Parameters: ***$uuid*** *string type is the uuid of the Folder*		

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    // const HOST = "http://localhost:8080/openkm/";
    const HOST = "http://192.168.1.11:8888/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $this->ws->folder->deleteFolder("56c4d33b-daad-4389-bfdc-ee8dd5273899");
            echo 'delete folder';
        } catch (Exception $e) {
            var_dump($e);
        }
```

```
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

renameFolder

Description:

Method	*Return values*	*Description*
renameFolder($uuid, $newName)	***void***	*Renames a folder.*

Parameters:

$uuid *string type is the uuid of the Folder*

$newName *string type is the new name for the Folder*

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            // Exists folder /okm:root/sdk/test
            $folder = $this->ws->folder->renameFolder("0b89e4ca-0019-4837-82a4-05bc05d83353", "renamedFolder");
            // Folder has renamed to /okm:root/sdk/renamedFolder
            echo 'rename Folder';
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
```

```
}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

moveFolder

Description:

Method	*Return values*	*Description*
moveFolder($uuid, $dstId)	***void***	*Moves a folder into 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***	*Copy 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*

$newName *string type is the new name for the Folder. Value is null, 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, the metadata, keywords, etc., of the folder are not copied.

Additionally:

- *When category parameter is true the original values of the categories will be copied.*
- *When keywords parameter is true the original values of the keywords will be copied.*
- *When propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When notes parameter is true the original values of the notes will be copied.*
- *When security parameter is true the original values of the security will be copied.*

Example:

```
<?php
```

```
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $folder = $this->ws->folder->extendedFolderCopy('4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5', '55102c84-e4f
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getContentInfo

Description:

Method	*Return values*	*Description*
getContentInfo($uuid)	***ContentInfo***	*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 in it.*
- *The number of documents in it.*
- *The number of records in it.*
- *The number of emails in 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 to /okm:trash/userId - the personal trash user location - but it is possible to directly purge any folder from the whole repository.

> *When a folder is purged, it will only be able to be restored from a previously created repository backup. The purge action permanently removes the folder 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();
?>
```

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, $language, $categories, $keywords, $propertyGroups, $notes, $properties)	***Folder***	*Creates a new folder from a template and returns a Folder object.*

*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, then these values are mandatory and must be set in the properties parameter.

Additionally:

- *When the categories 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.*

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; 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 from a node.*

Parameters:

$uuid *string type is the uuid of the document, folder, mail or record.*

$grpName *string type; 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 set 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, $grpName)	***array***	*Retrieves a list of all metadata group elements defined for a node.*
Parameters: ***$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 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. The grpName should be a valid metadata group name.*

$properties *array type. An array of properties.*

Before changing metadata, you ***should have the group added to 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 specify 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="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 if 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-1b
        } 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.*

The method can only be executed by superusers with ROLE_ADMIN membership.

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 is 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 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. The name of the property group.*

The method can only be executed by superusers with ROLE_ADMIN membership.

Example:

```
<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class Test {

    const HOST = "http://localhost:8080/openkm/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST);
    }

    public function test() {
        echo '<pre>';
        try {
            $this->ws->login(self::USER, self::PASSWORD);
            $fileName = dirname(__FILE__) . '/propertygroup/propertyGroups.xml';
            $this->ws->propertyGroup->registerPropertyGroupDefinition(file_get_contents($fileName), 'sdk4php');
            var_dump('Register Definition correct');
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->test();
?>
```

getPropertyGroupProperties

Description:

Method	*Return values*	*Description*
getPropertyGroupProperties($uuid, $grpName)	***array***	*Retrieves a list of all metadata group elements and 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:

Return

Method	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. 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 that 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 this is your case, you can use NodeProperties classes to retrieve the Object variables that 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 that 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" indicates 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 11-20, 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 this is your case, you can use NodeProperties classes to retrieve the Object variables that 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 being logged 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 your **http://your_domain/openkm/Preview**?docUrl=http://localhost/test.pdf*

*The **supported MIME types depend on your OpenKM configuration and OpenKM conversion service**.*

This component previews the most common file types:

- ***PDF, Microsoft Office** and **OpenOffice***
- ***Images like JPG, PNG, TIFF, MP4, MP3***
- ***Source code files like Java, PHP, HTML, SQL***

Preview parameters:

Parameter	*Type*	*Description*
docUrl	***String***	*The full URL of a document.* *We recommend using it for all 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.*

codUrl	***String***	*The full URL of a file containing code, for example .java, .php, etc. files.*
mpgUrl	***String***	*The full URL of an .mp4 or .mp3 file.*
pdfUrl	***String***	*The full URL of a PDF document. Required by the Flexpaper previewer.*
file	***String***	*The full URL of a PDF document. Required by the PDF.js previewer.* *PDF.js also requires the **pdfUrl** parameter.*
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.*

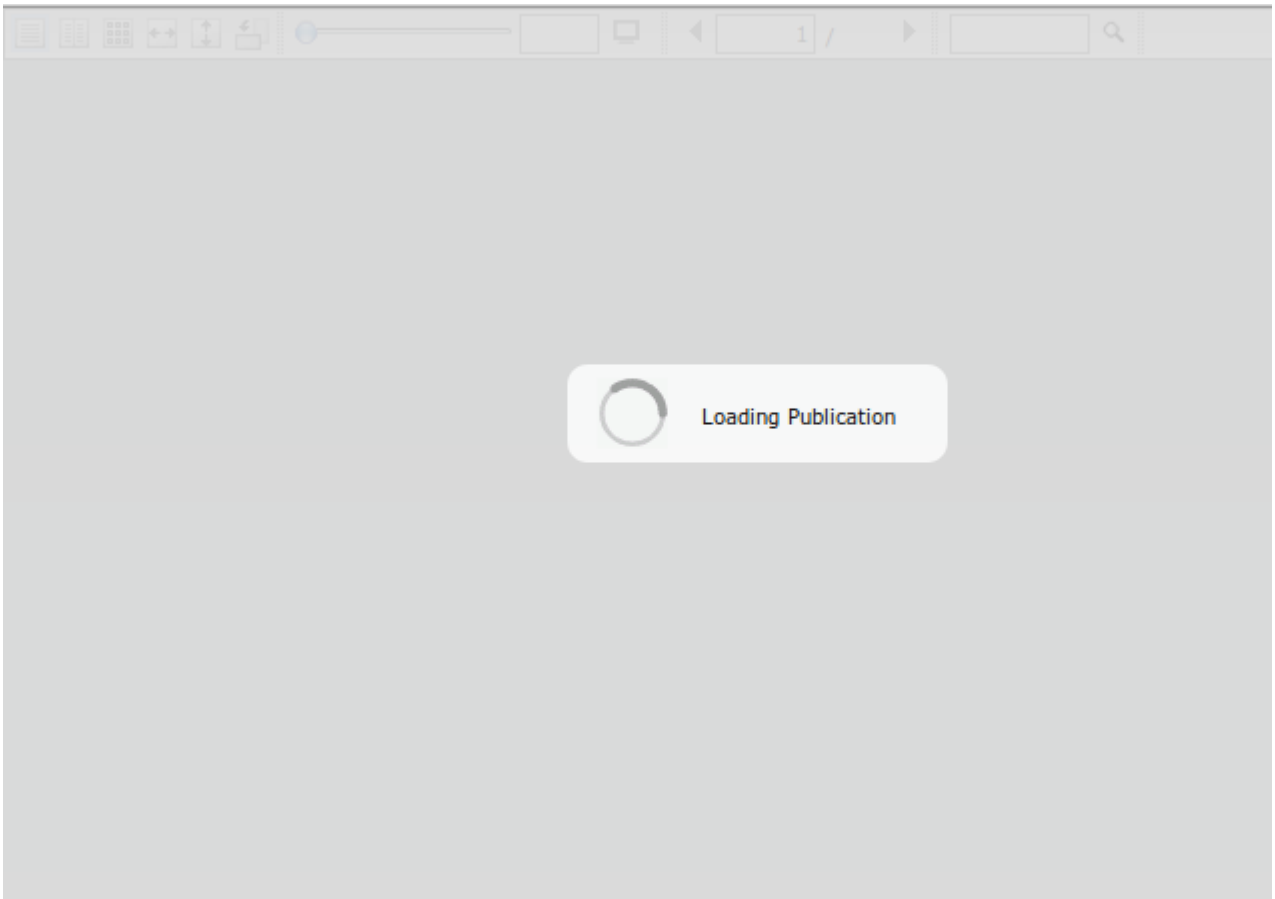
The Java code section below can provide some ideas about it:

```
/**
 * 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 at 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 property group.* • *Shows the number of documents and folders, and the size of a given path.* • *Gets the path by UUID.*
Script - Compact documents with size equal to 0	*Finds all documents with size=0 and compacts their version history to the latest version.*
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 within the first-level folders (okm:root children folders).*
Script - Exporter	*Performs 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 folder without first moving files to the trash.*
Script - Purge all users' trash	*Example of how to clean all users' trash.*
Script - Read PDF form fields	*Shows how to read PDF form fields.*

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

Script - Recursive character renaming	*Renames a character in all folders or documents from a specified initial path.*
Script - Recursive repository traversal	*Navigates across all repository nodes recursively.*
Script - Recursive change security	*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;
```

```
OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
String path = okmRepository.getNodePath(null, "162735a1-661c-4555-bf53-4e9115f38c62");
print(path)
```

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

Script - Compact documents with size equal to 0

Find all documents with size=0 and compact their version history to the latest version.

```
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.*
- *For 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 from 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 with a certain depth.

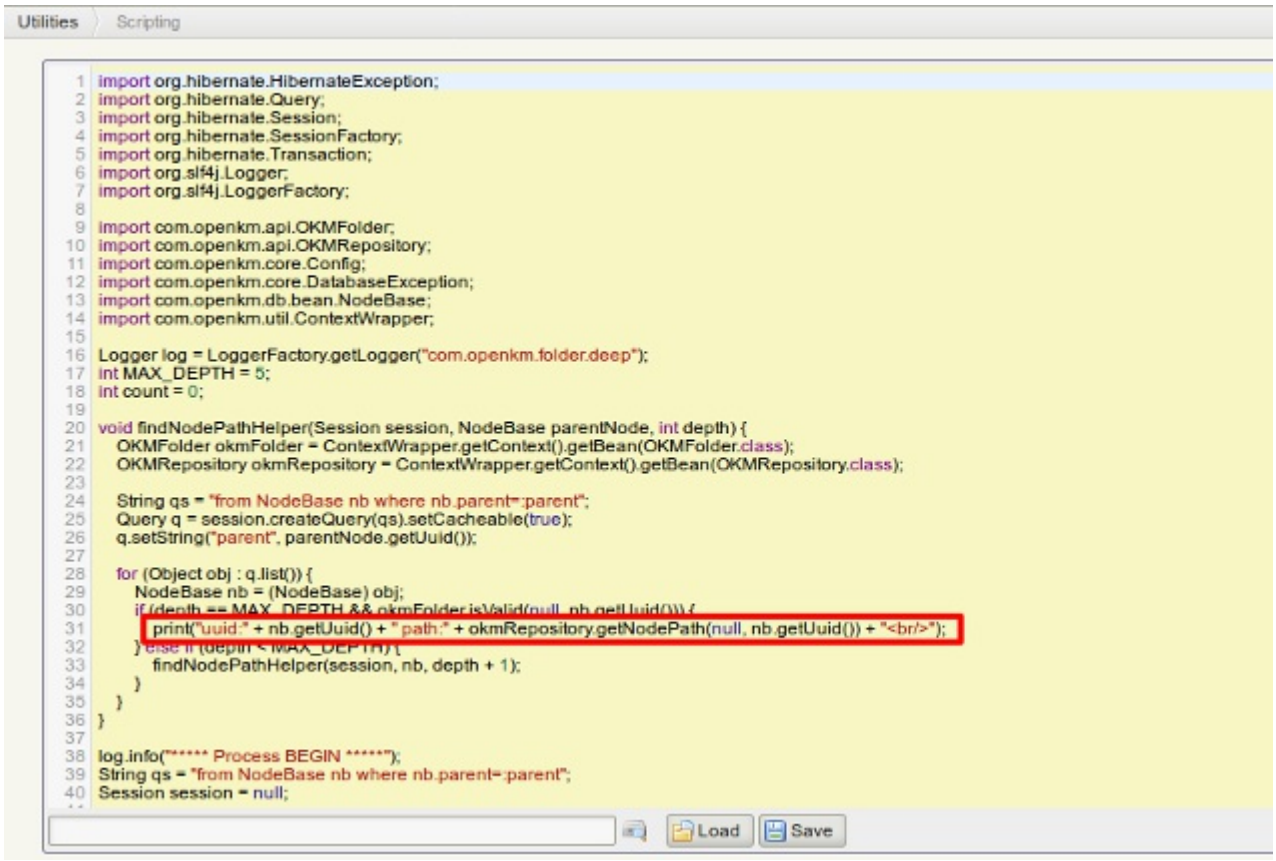
*The variable **MAX_DEPTH=5** indicates the maximum folder depth.*

```
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

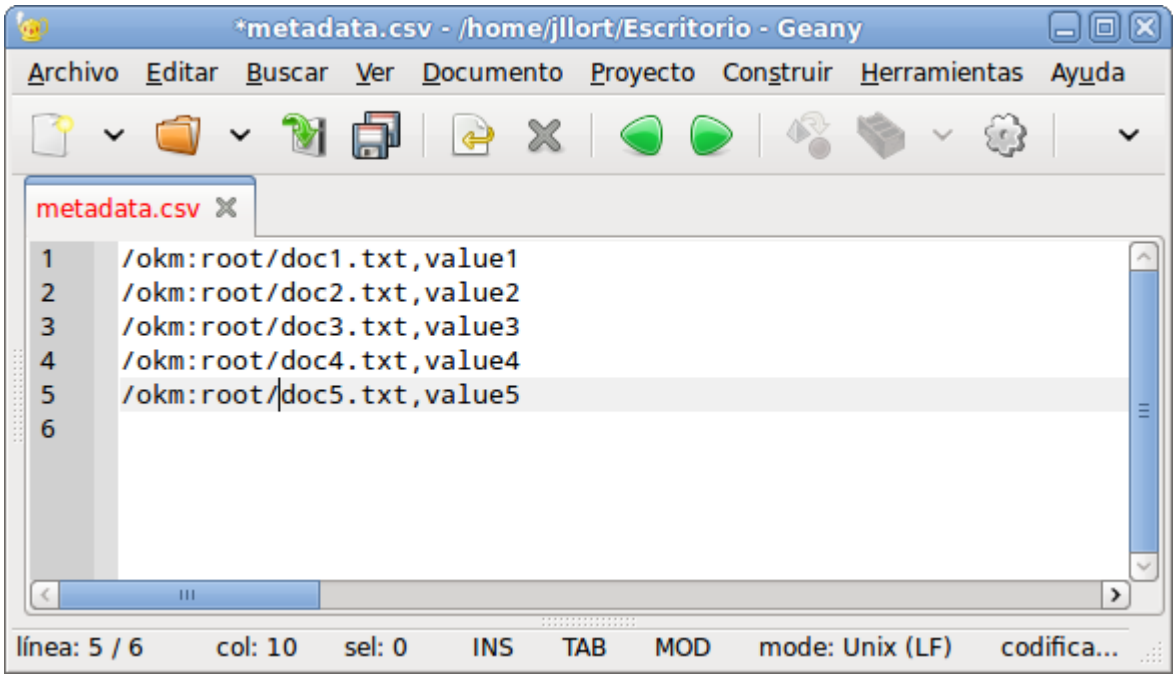
Script error
Script result
Script output
uuid:e18393ad-912f-4b0a-8095-596456d52ae8 path:/okm:root/test 1.2/test2/test3/test4/test5 uuid:c0b84029-da90-472a-80e5-44819b2a47f3 path:/okm:root/test 1.1/test2/test3/test4/test5 uuid:5dbc701b-658e-4e4b-ba8b-710a7e831fac path:/okm:root/test 1.3/test2/test3/test4/test5 uuid:70606548-586e-4a10-94c1-7b32d7734f4d path:/okm:trash/okmAdmin/test 1.4/test2/test3/test4

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's 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:

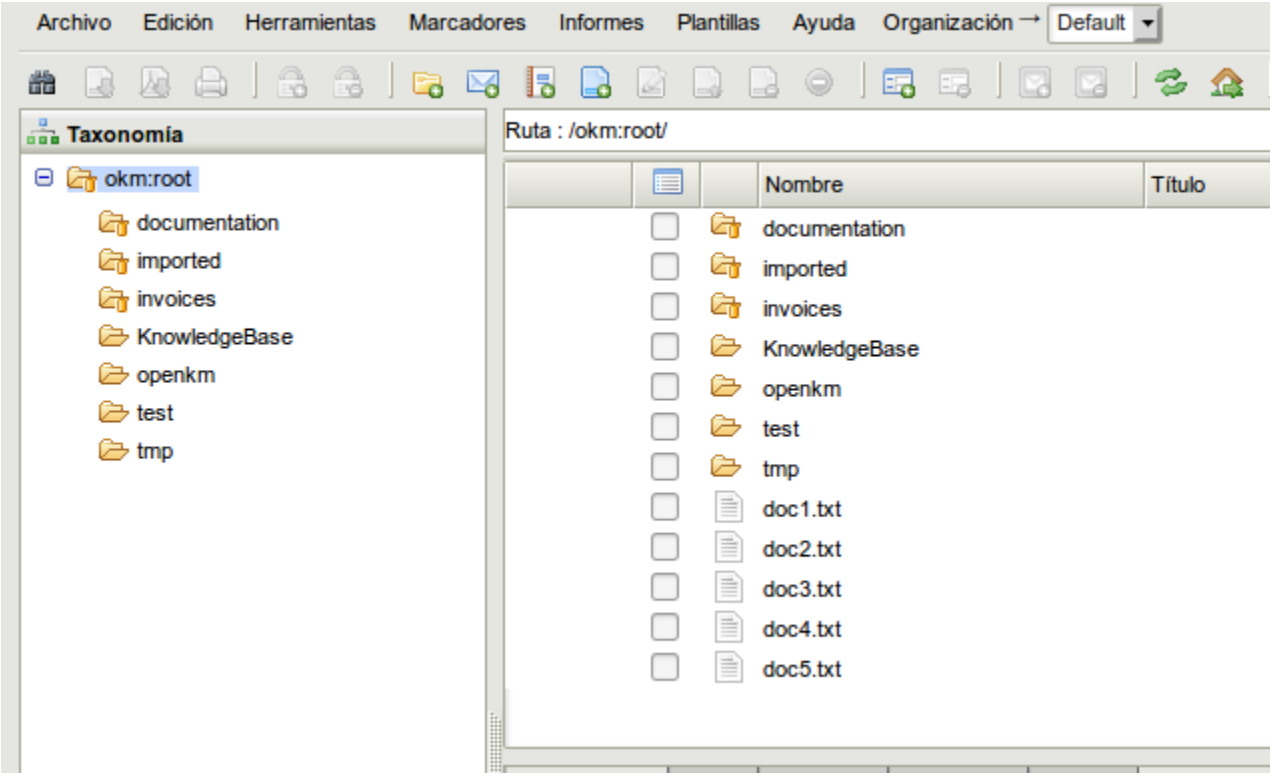
```
import java.io.FileReader;
import java.io.Reader;
import java.util.Collection;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

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>");
```

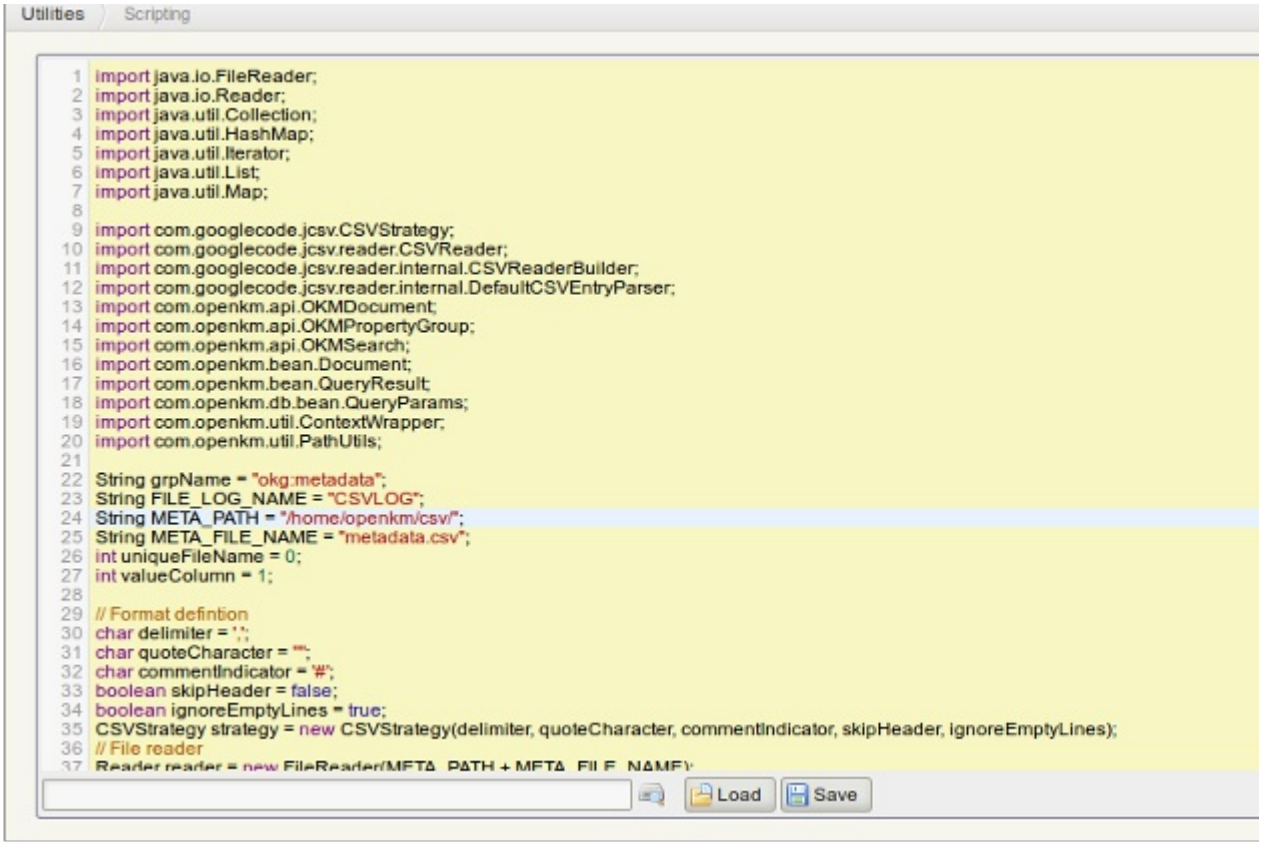
Image

Register the metadata group

Repository view

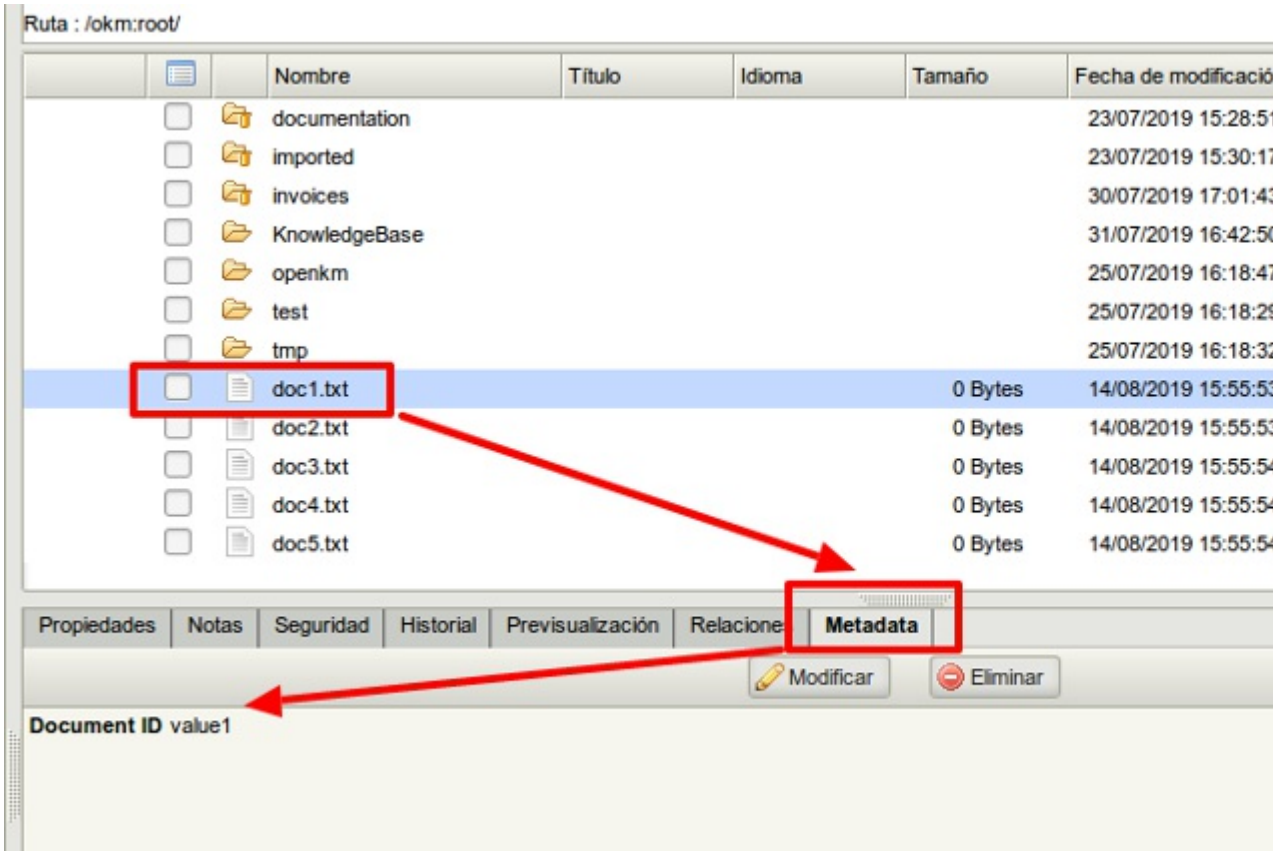

Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
imported
invoices
KnowledgeBase
openkm
test
tmp
Ruta : /okm:root/
Nombre
Título
documentation
imported
invoices
KnowledgeBase
openkm
test
tmp
doc1.txt
doc2.txt
doc3.txt
doc4.txt
doc5.txt

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

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 the 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();
}
```

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

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 and documents starting from a specified 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 a low level

This is an administrator script that should be used only when it is necessary to propagate security changes under a folder that has 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 to all 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's unique identifier.*
- *The variable **MAX_DEPTH** sets how deep the script should advance into subfolders (note that 0 indicates the UUID folder).*
- *The variables **roleName** and **roleGrant** are arrays 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);
    }
```

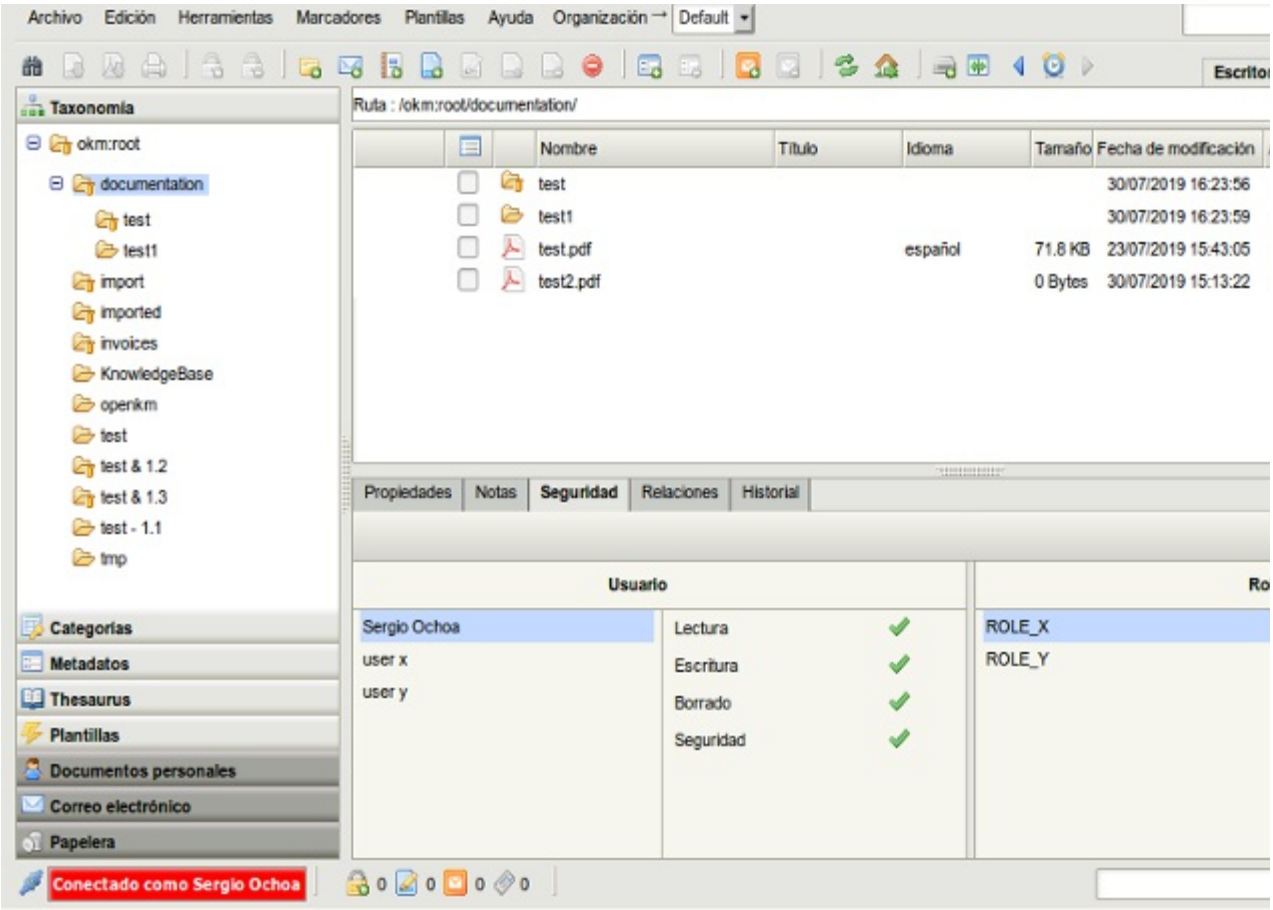
```
    if (level < MAX_DEPTH) {
        String sqlfindFolderChildren = "SELECT NBS_UUID FROM OKM_NODE_BASE WHERE NBS_PARENT=? an

        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());
}
```

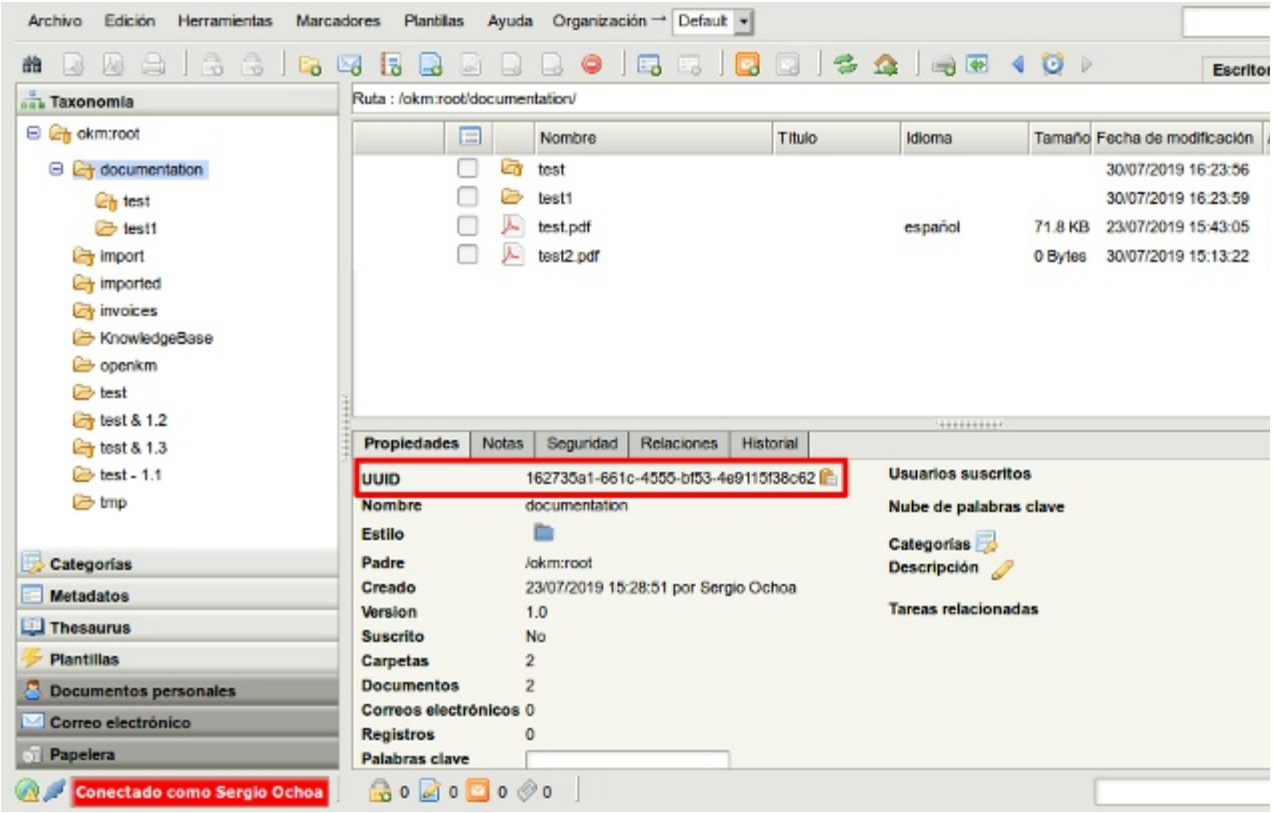
After the script execution, we must reindex the Lucene indexes because the security changes have been made at the database level, and this is necessary to maintain integrity with the search engine.

Images

Initial security status

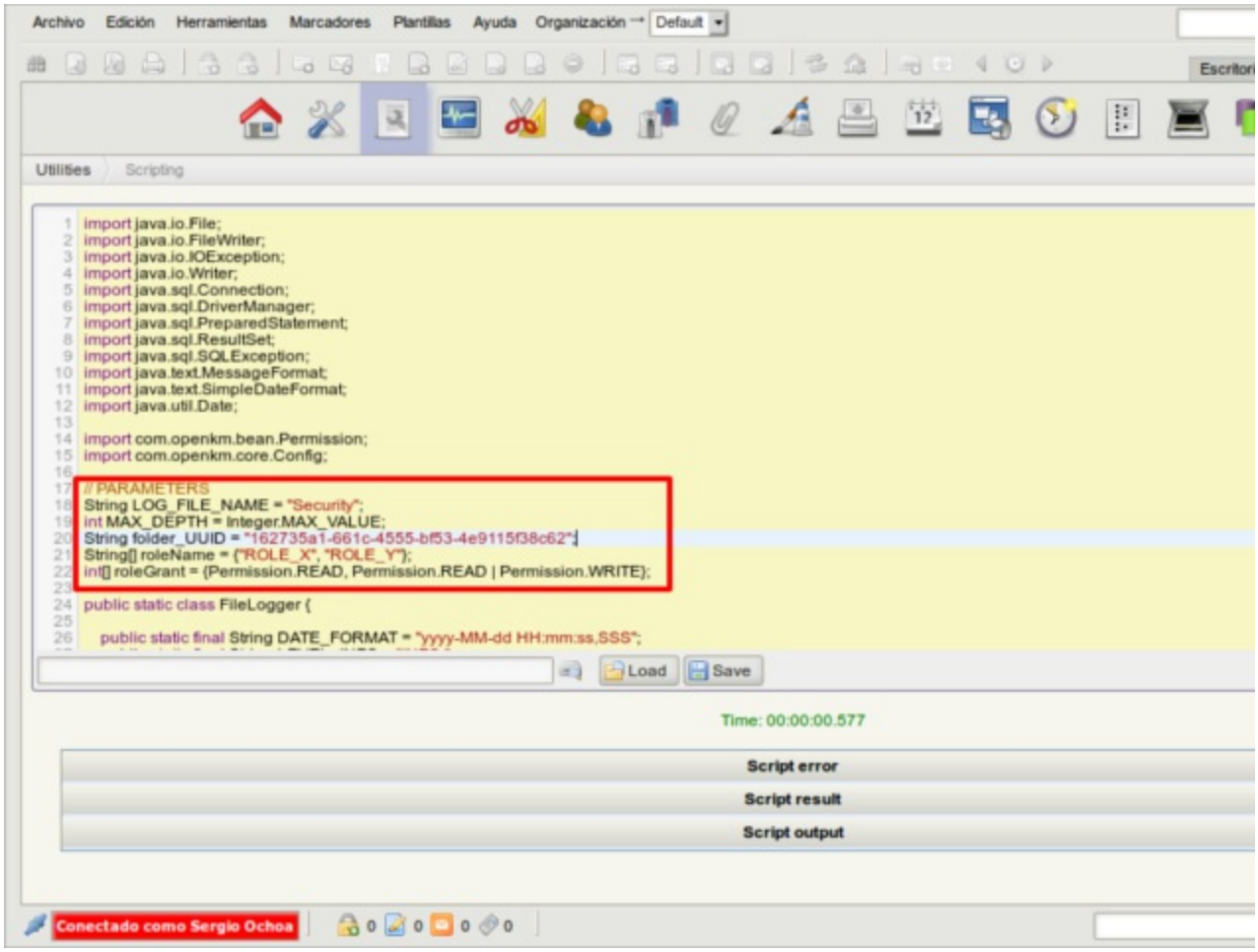


UUID of the folder



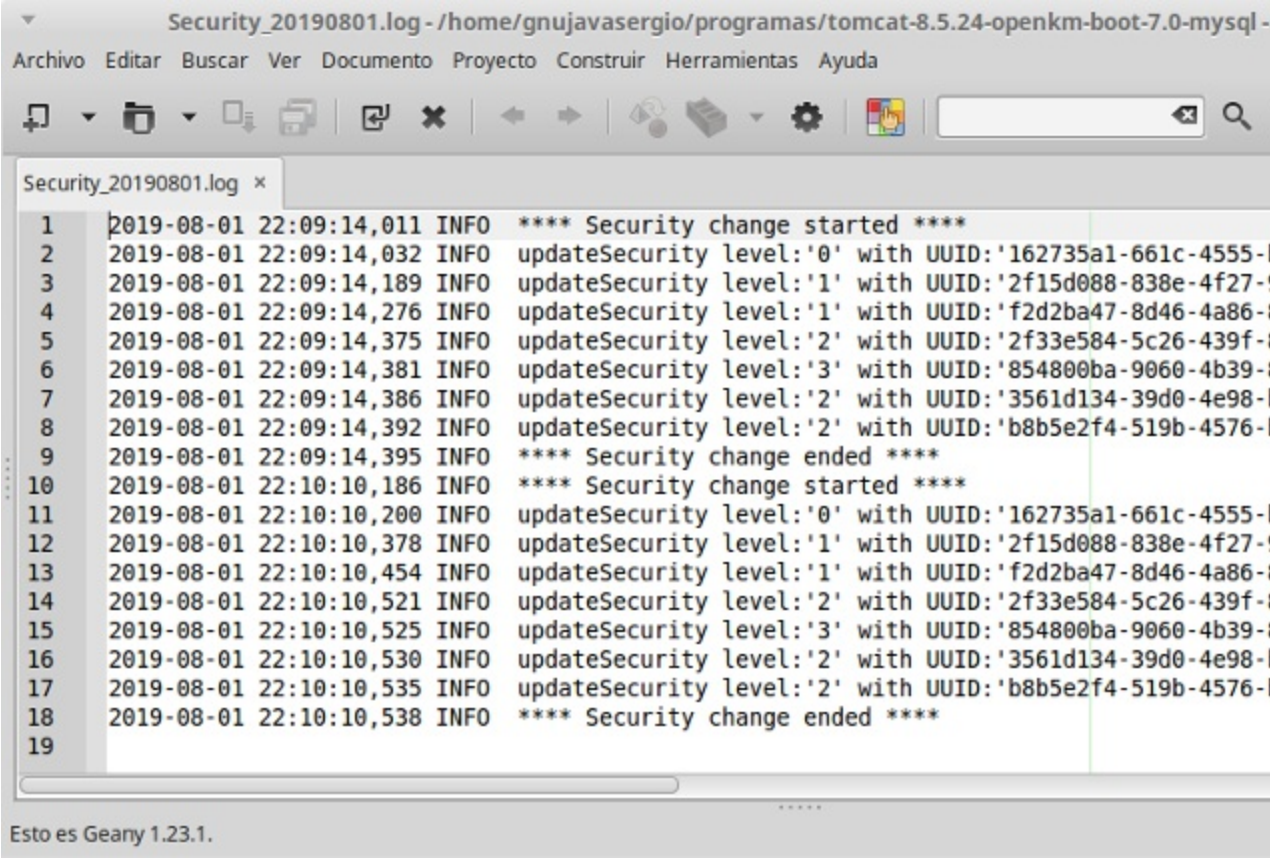
Execution

Change the script parameters to your own.



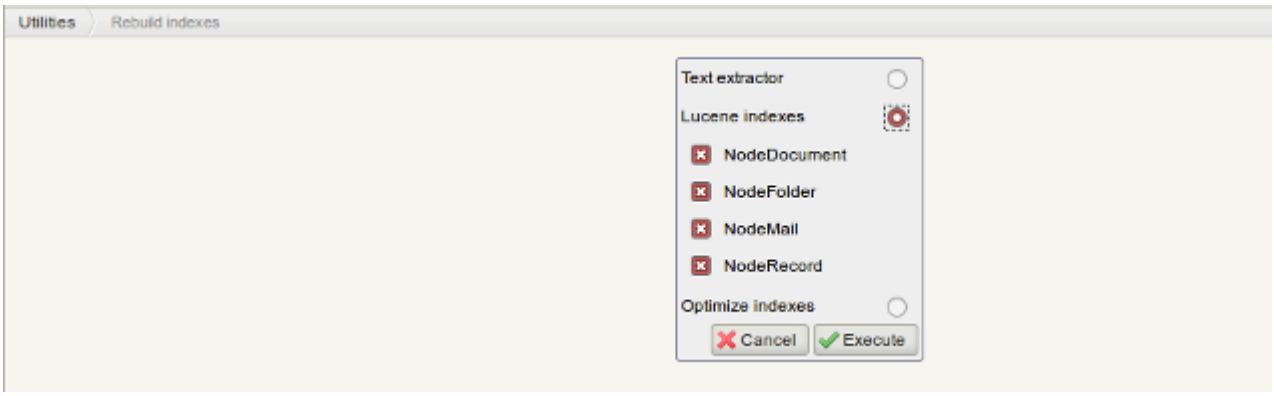
Log

Consider taking a look at the log file to see the progress of the process and check for errors.



Rebuild Lucene indexes

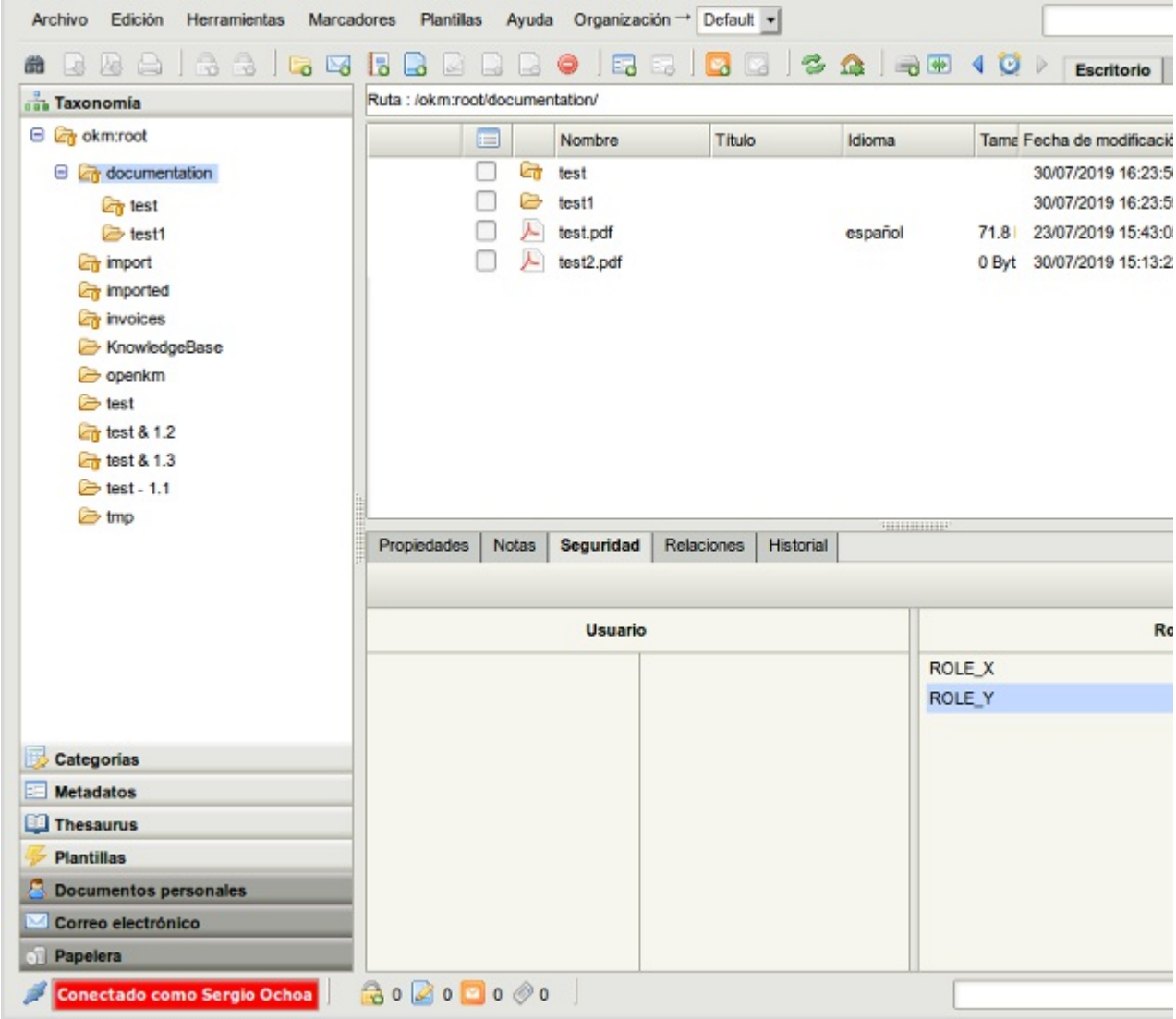
Because the security changes were made at the database level, it is necessary to reindex the Lucene indexes to maintain integrity with the search engine.



Utilities > Rebuild indexes > Rebuild Lucene indexes

- Mode: FlushToIndexes
- System into maintenance mode
- Calculate NodeDocument
- Number of NodeDocument: 42
- Calculate NodeFolder
- Number of NodeFolder: 46
- Calculate NodeMail
- Number of NodeMail: 0
- Calculate NodeRecord
- Number of NodeRecord: 0
- Rebuilding indexes
- Indexing NodeDocument
  ..... 30% . * . 40% . * . 50% ..... 80% . * . 90% . * . 100%
- Indexing NodeFolder
  .... 10% . * ..... 30% ..... * . 50% .... 60% . * ..... 80% ..... * . 100%
- Indexing NodeMail
- Indexing NodeRecord
- System out of maintenance mode
- Index rebuilding completed!
- Elapsed time: 00:00:01.373

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 the values of a metadata group.*

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, automation is executed. The automation script creates a new unique numeric identifier using the metadata database 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.*

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.*
- *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 that generates **okp:data.id** and renames the document based on the mask **projectCode-autonumericId-clientCode-description.documentExtension**.*

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 for use 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 an automation rule

Create an 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's file system.*
- *Files are periodically imported by a crontab task into 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 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

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-

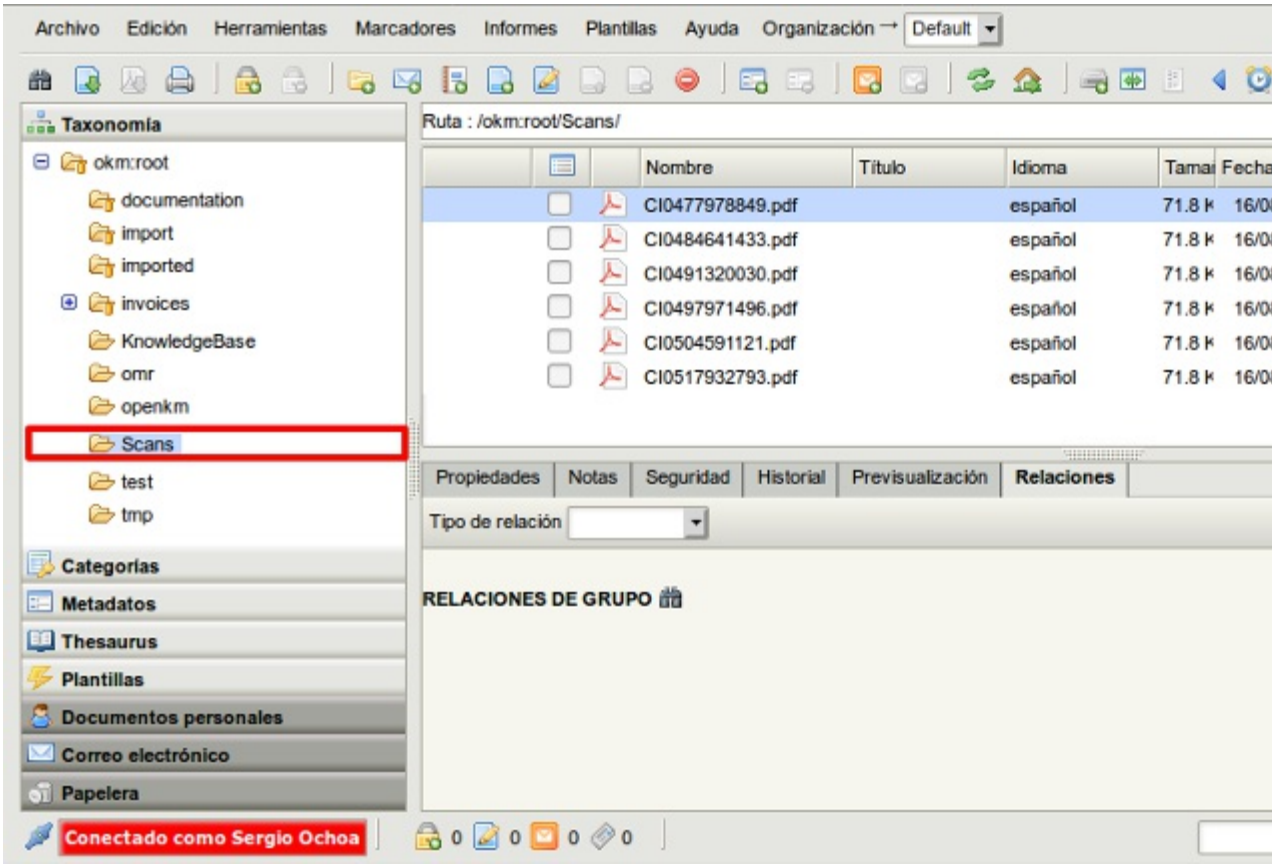
Showing 1 to 15 of 17 entries

Files on the application server

CI0477978849.pdf	244,2 KB Adobe Acrobat Document	29/09/11
CI0484641433.pdf	270,3 KB Adobe Acrobat Document	29/09/11
CI0491320030.pdf	276,4 KB Adobe Acrobat Document	29/09/11
CI0497971496.pdf	279,0 KB Adobe Acrobat Document	29/09/11
CI0504591121.pdf	278,9 KB Adobe Acrobat Document	29/09/11
CI0517932793.pdf	305,1 KB Adobe Acrobat Document	29/09/11

Result

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


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
import
imported
invoices
KnowledgeBase
omr
openkm
Scans
test
tmp
Categorías
Metadatos
Thesaurus
Plantillas
Documentos personales
Correo electrónico
Papelera
Conectado como Sergio Ochoa
Ruta : /okm:root/Scans/
Nombre
Título
Idioma
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 text between the characters "**(**" and "**)**" present in the file name.*
- *Files are stored in basePath/year/month (when year or month folders are not present, they are automatically created).*
- *At the end, it sends an email with the 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;
```

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

```
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 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****.*

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:* s • *Label:* Octubre, *Value:* octu • *Label:* Noviembre, *Value:* n • *Label:* Diciembre, *Value:* di

Crontab task

Crontab

Show 15 entries

Name	Expression	Class Name	Last Begin	
Antivirus Checker Worker	0 */5 * * * *	com.openkm.plugin.cron.AntivirusChecker	18-08-2019 11:45:00	18-
Check Disk Space	0 0 0 * * *	com.openkm.plugin.cron.CheckDiskSpace	18-08-2019 00:00:00	18-
Check Repository Integrity	0 0 0 * * SAT	com.openkm.plugin.cron.CheckRepositoryIntegrity		
Dashboard Activity Purge	0 0 0 * * *	com.openkm.plugin.cron.DashboardActivity	18-08-2019 00:00:00	18-
Document Importer With Metadata	0 5 * * * *	com.openkm.plugin.cron.DocumentImporterWithMetadata	18-08-2019 11:27:28	18-
Process Pending Task	0 */5 * * * *	com.openkm.plugin.cron.ProcessPendingTask	18-08-2019 11:45:00	18-
Purge Empty Datastore Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyDatastoreDirectories	18-08-2019 00:00:00	18-
Purge Empty Extraction Directories	0 0 0 * * *	com.openkm.plugin.cron.PurgeEmptyExtractionDirectories	18-08-2019 00:00:00	18-
Purge Old Trash Nodes	0 0 0 * * *	com.openkm.plugin.cron.PurgeOldTrashesNodes	18-08-2019 00:00:00	18-
Purge Temporal Printing Files	0 0 0 * * *	com.openkm.plugin.cron.PurgeTemporalPrintingFiles	18-08-2019 00:00:01	18-
Purge Tomcat Logs	0 0 0 * * *	com.openkm.plugin.cron.PurgeTomcatLogs	18-08-2019 00:00:00	18-
Repository Info	0 0 0 * * *	com.openkm.plugin.cron.RepositoryInfo	18-08-2019 00:00:00	18-
Send Grouped Notifications	0 0 0 * * *	com.openkm.plugin.cron.SendGroupedNotifications	18-08-2019 00:00:00	18-
System Document Importer	0 5 * * * *	com.openkm.plugin.cron.SystemDocumentImporter	18-08-2019 02:05:02	18-
Text Extractor Issues	0 0 0 * * *	com.openkm.plugin.cron.TextExtractorIssues	18-08-2019 00:00:00	18-

Showing 1 to 15 of 17 entries

Files 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

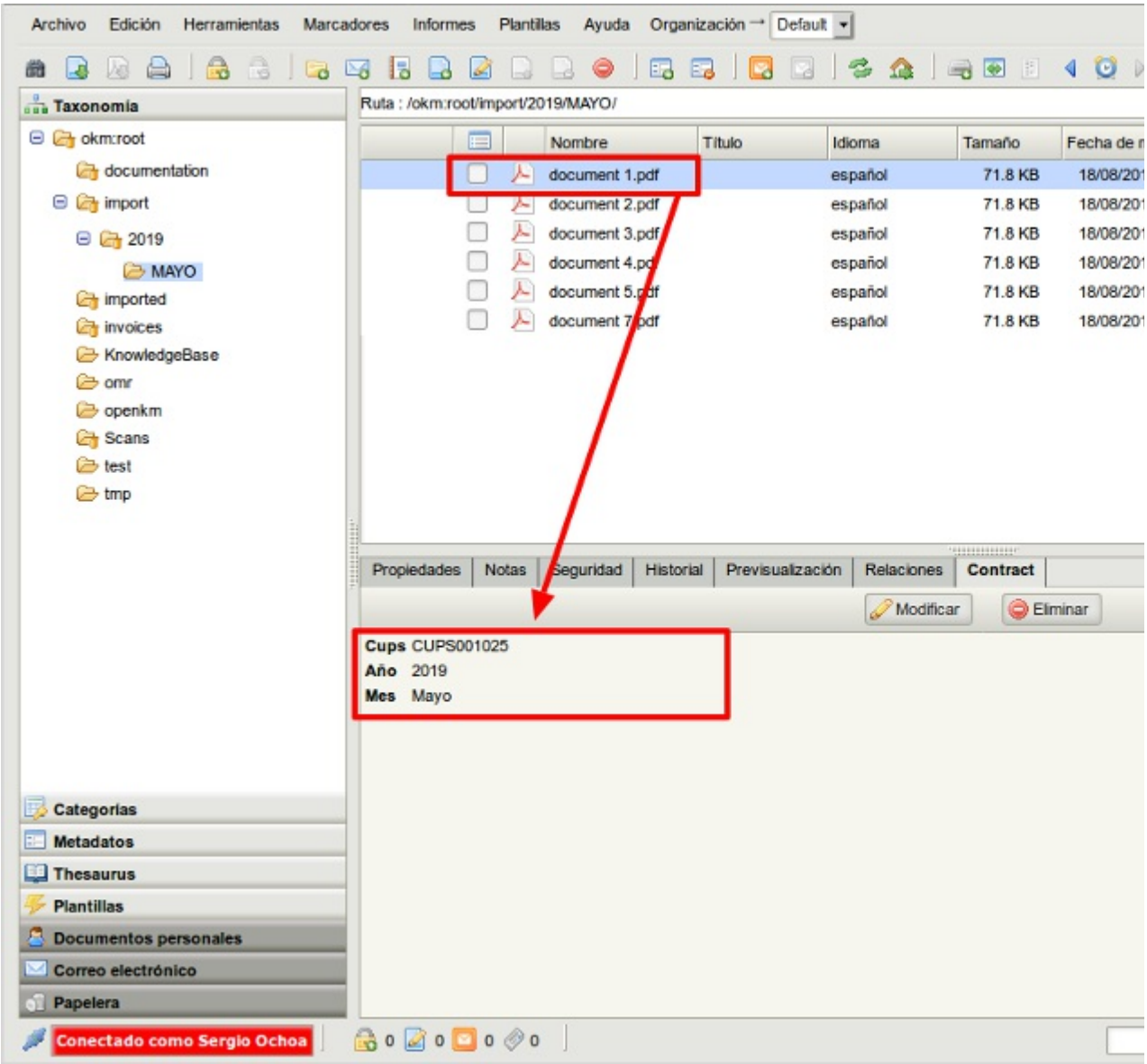
Created in 2016 by OpenKM. OpenKM documentation is provided under OpenKM documentation license.

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


Archivo
Edición
Herramientas
Marcadores
Informes
Plantillas
Ayuda
Organización
Default
Taxonomía
okm:root
documentation
import
2019
MAYO
imported
invoices
KnowledgeBase
omr
openkm
Scans
test
tmp
Ruta : /okm:root/import/2019/MAYO/
Nombre
Título
Idioma
Tamaño
document 1.pdf
document 2.pdf
document 3.pdf
document 4.pdf
document 5.pdf
document 7.pdf
español
71.8 KB
Propiedades
Notas
Seguridad
Historial
Previsualización
Relaciones
Contract
Modificar
Eliminar
Cups CUPS001025
Año 2019
Mes Mayo
Categorías
Metadatos
Thesaurus
Plantillas
Documentos personales
Correo electrónico
Papelera
Conectado como Sergio Ochoa

Crontab sample - Generate an 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)) {
```

```
    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.*
- *In the* ***plugins table*** *the new plugin will be shown.*

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>
```

```
            <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).getTextC
                                            .trim();
                                    String value = fieldElement.getElementsByTagName(ATTRIBUTE_VALUE).item(0).getTextC
                                            .trim();
```

```
                    if (name.toLowerCase().equals(FIELD_NAME_FOLIO)) {
                        folio = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_RUT)) {
                        rut = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_PLACA_PATENTE)) {
                        placaPatente = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_NOMBRES)) {
                        nombres = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_APELLIDO_PATERNO)) {
                        apellidoPaterno = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_APELLIDO_MATERNO)) {
                        apellidoMaterno = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_ANO)) {
                        ano = value;
                    } else if (name.toLowerCase().equals(FIELD_NAME_TIPO_DOCUMENTO)) {
                        tipoDocumento = value;
                    }
                }
            }
        }

        if (fileName != null && !fileName.equals("")) {
            QueryParams queryParams = new QueryParams();
            queryParams.setDomain(QueryParams.DOCUMENT);
            queryParams.setName(fileName);
            List<QueryResult> results = okmSearch.find(token, queryParams);

            if (results.size() == 1) {
                for (QueryResult queryResult : results) {
                    if (queryResult.getNode() != null && queryResult.getNode() instanceof Document) {
                        log.info(fileName + " -ok");
                        boolean found = false;

                        for (PropertyGroup group : okmPropertyGroup.getGroups(token,
                                queryResult.getNode().getPath())) {
                            if (group.getName().equals("okg:xmldata")) {
                                found = true;
                            }
                        }

                        Map<String, String> properties = new HashMap<String, String>();
                        properties.put("okp:xmldata.folio", folio);
                        properties.put("okp:xmldata.rut", rut);
                        properties.put("okp:xmldata.placa", placaPatente);
                        properties.put("okp:xmldata.nombres", nombres);
                        properties.put("okp:xmldata.apellido_padre", apellidoPaterno);
                        properties.put("okp:xmldata.apellido_madre", apellidoMaterno);
                        properties.put("okp:xmldata.year", ano);
                        properties.put("okp:xmldata.documento", tipoDocumento);
                        if (!found) {
                            okmPropertyGroup.addGroup(token, queryResult.getNode().getPath(), "okg:xmldata",
                                    properties);
                        } else {
                            okmPropertyGroup.setProperties(token, queryResult.getNode().getPath(), "okg:xmldata
                                    properties);
                        }
                    }
                }
            } else if (results.size() > 1) {
                log.error(fileName + " - error");
            } else {
                log.error(fileName + " - not found");
            }
        }
    }
}
```

```
                xmlFile.delete();
            }
        }
    }

    /**
     * importFiles
     */
    public void importFiles() throws UnsupportedMimeTypeException, VirusDetectedException, ItemExistsException,
            PathNotFoundException, AccessDeniedException, RepositoryException, DatabaseException, AutomationEx
            PromotedAsRecordLockException, PrincipalAdapterException, IOException {
        // Loading files
        File folder = new File(SYSTEM_FOLDER + "/Output");
        File[] listOfFiles = folder.listFiles();
        for (int i = 0; i < listOfFiles.length; i++) {
            File file = listOfFiles[i];
            if (file.isFile() && file.getName().toLowerCase().endsWith(".pdf")) {
                Document doc = new Document();
                doc.setPath(OPENKM_FOLDER + "/" + file.getName());
                FileInputStream fis = new FileInputStream(file);
                doc = okmDocument.create(token, doc, fis);
                file.delete();
            }
        }
    }
}
```

Register a new plugin

- *To install the new plugin, **create a JAR file** and **copy it into your $TOMCAT/plugins** folder—more information at [Creating a JAR file with Eclipse](#).*
- *Go to **Administration** > **Utilities** > **Plugins** and click, at the top right, the* *Reload plugins button.*
- *In **Plugins**, the 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

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

Metadata contracts

Group label		Group name		
Contract		okg:contract		
Label	**Name**	**Width**	**Height**	**Element**
Cups	okp:contract.cups	200px	25px	Input
Año	okp:contract.year	200px	25px	Input
Mes	okp:contract.month	150px	25px	Select

Readonly: false
Data:
Type: text
Value:
Description:

Readonly: false
Data:
Type: text
Value:
Description:

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

home openkm logfile

2013-04-16 - IndexLog.xml	517,2 KB documento XML
2013-04-15 - IndexLog.xml	851,5 KB documento XML

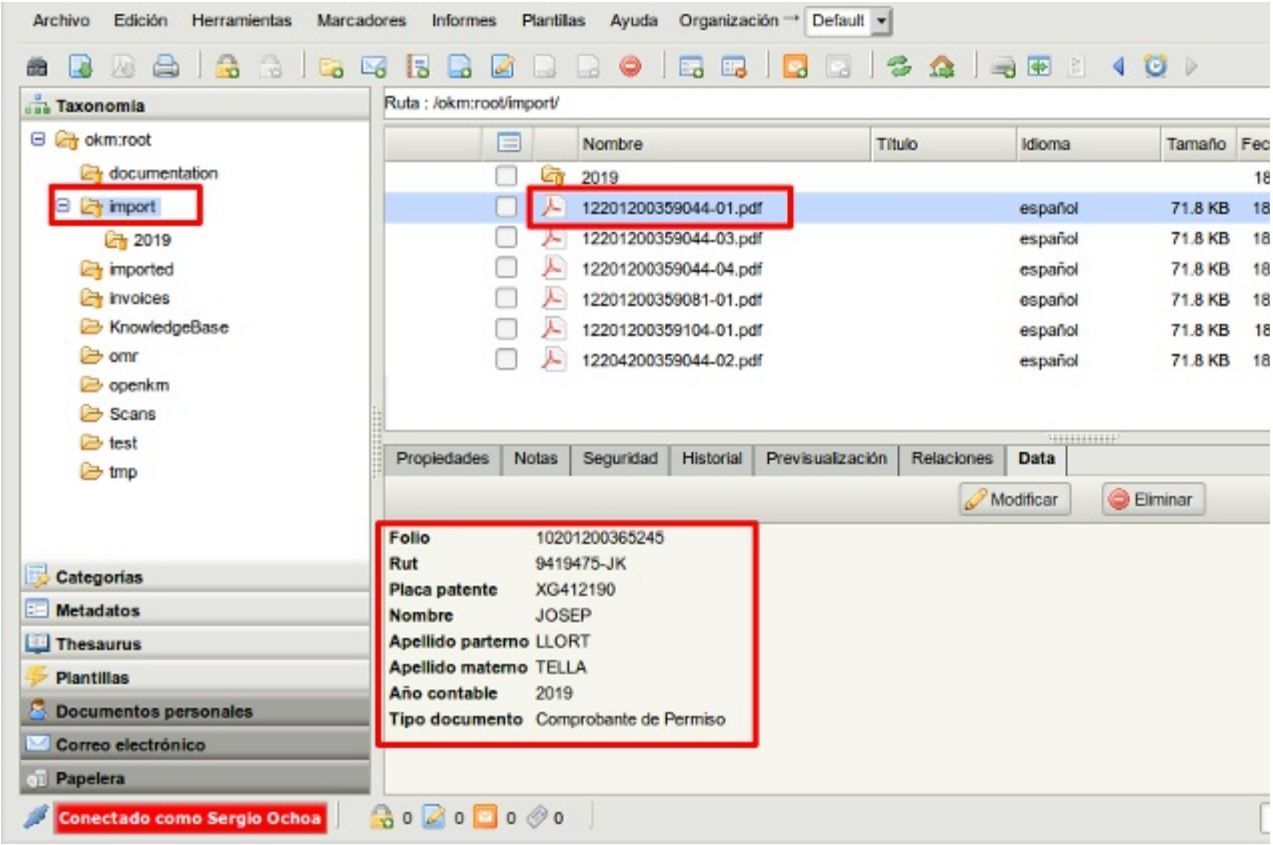
Output folder

home openkm Output

12201200359104 - 01.pdf	51,2 KB Adobe Acrobat Document
12201200359081 - 01.pdf	53,5 KB Adobe Acrobat Document
12201200359044 - 04.pdf	415,1 KB Adobe Acrobat Document
12201200359044 - 03.pdf	31,6 KB Adobe Acrobat Document
12201200359044 - 02.pdf	39,3 KB Adobe Acrobat Document
12201200359044 - 01.pdf	58,8 KB Adobe Acrobat Document

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
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
español
71.8 KB
12201200359044-03.pdf
12201200359044-04.pdf
12201200359081-01.pdf
12201200359104-01.pdf
12204200359044-02.pdf
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 (previous) and post (posterior). For example, a pre document creation event is executed before document creation, and several event vars can then be changed to modify the parent folder, document name, among others. In the case of a post document creation event, it is executed after document creation. In this case, there are no vars associated with the event; you'll get a NodeDocument object to apply any changes directly through the OpenKM API.

Available environment variables

Variable	*Object class*	*Description*
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*
	NodeBase	

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

AutomationUtils.NODE_BASE	• *NodeDocument* • *NodeFolder* • *NodeMail* • *NodeRecord* *These nodes extend NodeBase.*	*Node base*
AutomationUtils.NODE_BASE_LIST	*List<NodeBase>* • *NodeDocument* • *NodeFolder* • *NodeMail* • *NodeRecord* *These nodes extend 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*
AutomationUtils.MAIL_REPLY	*String[]*	*Mail reply*

AutomationUtils.MAIL_SENT_DATE	*Calendar*	*Mail sent date*
AutomationUtils.MAIL_SUBJECT	*String*	*Mail subject*
AutomationUtils.MAIL_TO	*String[]*	*Mail to*
AutomationUtils.MIME_TYPE	*String*	*MIME type*
AutomationUtils.NAME	*String*	*Name*
AutomationUtils.PROPERTY_GROUP_NAME	*String*	*Property group name*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*Map<String, String>*	*Property group properties*
AutomationUtils.PROPERTY_GROUPS	*Set<NodeProperty>*	*List of property groups (metadata groups)*
AutomationUtils.RECORD_NODE	*NodeRecord*	*Record node*
AutomationUtils.TEXT_EXTRACTED	*String*	*Extracted document text*
AutomationUtils.LANGUAGE_DETECTED	*String*	*Detected language*
AutomationUtils.UPLOAD_RESPONSE	*String*	*Upload response*
AutomationUtils.TITLE	*String*	*Title*
AutomationUtils.WIKIPAGE	*wikiPage*	*The wiki page*
AutomationUtils.NODE_CLASS	*Long*	*The NodeClass ID*
AutomationUtils.NEW_NODE_CLASS		

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

	Long	*The new NodeClass*
AutomationUtils.OLD_NODE_CLASS	*Long*	*The old NodeClass*
AutomationUtils.TASK_ID	*Long*	*The Task ID*
AutomationUtils.TASK_OBJECT	*TaskManagerTask*	*Task object*
AutomationUtils.USER	*com.openkm.bean.CommonUser*	*User*
AutomationUtils.DOCUMENT_SIZE	*Long*	*Document size*

Document

Create Document

At Pre	
Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*
AutomationUtils.CREATION_DATE	*No*
AutomationUtils.NAME	*Yes*
AutomationUtils.MIME_TYPE	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.NOTES	*Yes*
AutomationUtils.NODE_CLASS	*Yes*

AutomationUtils.DESTINATION_NODE ***At Pre***	*Yes*
AutomationUtils.PROPERTY_GROUPS	*Yes*
AutomationUtils.TITLE	*Yes*
AutomationUtils.WIKIPAGE	*Yes*
AutomationUtils.DOCUMENT_SIZE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_FILE	*No*

Update Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Delete Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Rename Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Purge Document

At Pre	
Variable	***For changing***

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

AutomationUtils.DOCUMENT_NODE	*No*
At Pre	
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Move Document

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

Restore Version Document

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*

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

At Pre
At post

Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*

Download Document from UI

At Pre

Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*
AutomationUtils.DOCUMENT_NAME	*Yes*
AutomationUtils.DOCUMENT_NODE	*No*

At post

Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_VERSION	*No*
AutomationUtils.DOCUMENT_NAME	*No*

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

AutomationUtils.DOCUMENT_NODE ***At Pre***	*No*

Download Document from UI for preview

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_NAME	*Yes*
AutomationUtils.DOCUMENT_VERSION	*No*
At post	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*
AutomationUtils.DOCUMENT_NAME	*No*
AutomationUtils.DOCUMENT_VERSION	*No*

Folder

Create Folder

At Pre	
Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*

AutomationUtils.CREATION_DATE ***At Pre***	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.NAME	*Yes*
AutomationUtils.NOTES	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.PROPERTY_GROUPS	*Yes*
AutomationUtils.WIKIPAGE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Delete Folder

The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***

At Pre	
AutomationUtils.FOLDER_NODE	*No*

Purge Folder

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Move Folder

> *The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Rename Folder

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Export Folder as ZIP

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

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

Export Folder as JAR

At Pre	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.FOLDER_NODE	*No*

Mail

Create mail

At Pre	
Variable	***For changing***
AutomationUtils.CATEGORIES	*Yes*
AutomationUtils.CREATION_DATE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.MAIL_FROM	*No*
AutomationUtils.MAIL_REPLY	*No*
AutomationUtils.MAIL_TO	*No*

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

AutomationUtils.MAIL_CC ***At Pre***	*No*
AutomationUtils.MAIL_BCC	*No*
AutomationUtils.MAIL_SENT_DATE	*No*
AutomationUtils.MAIL_RECEIVED_DATE	*No*
AutomationUtils.MAIL_SUBJECT	*No*
AutomationUtils.MAIL_CONTENT	*No*
AutomationUtils.NAME	*Yes*
AutomationUtils.KEYWORDS	*Yes*
AutomationUtils.MIME_TYPE	*No*
AutomationUtils.TITLE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Move Mail

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

At Pre	
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Download mail from UI

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Import mail attachments

At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Delete 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 from them.

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Purge Mail

The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.

At Pre	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.MAIL_NODE	*No*

Record

Create record

At Pre	
Variable	***For changing***
AutomationUtils.DESTINATION_NODE	*Yes*
AutomationUtils.NAME	*Yes*
AutomationUtils.NODE_CLASS	*Yes*
AutomationUtils.TITLE	*Yes*
At post	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*

Delete record

The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***

AutomationUtils.RECORD_NODE	*No*
At Pre	

Purge record

> *The event is only triggered on the root node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
At post	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*

Move Record

> *The event is only triggered on the root node and the destination node. If the node has subnodes, an event will not be generated from them.*

At Pre	
Variable	***For changing***
AutomationUtils.RECORD_NODE	*No*
AutomationUtils.DESTINATION_NODE	*Yes*
At post	
Variable	***For changing***

At Pre	
AutomationUtils.RECORD_NODE	*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*

Set property group

At Pre	
Variable	***For changing***

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

AutomationUtils.NODE_BASE	*No*
At Pre	
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*Yes*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
AutomationUtils.PROPERTY_GROUP_PROPERTIES	*No*

Remove property group

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.PROPERTY_GROUP_NAME	*No*

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

At Pre

Auth

Create user

At Pre	
Variable	***For changing***
AutomationUtils.USER	*No*
At post	
Variable	***For changing***
AutomationUtils.USER	*No*

User login

At post	
Variable	***For changing***
AutomationUtils.USER	*No*

User logout

At Pre	
Variable	***For changing***
AutomationUtils.USER	*No*

Note

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

Create Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_TEXT	*No*
At post	
Variable	***For changing***
AutomationUtils.NOTE_UUID	*No*

Update Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_TEXT	*No*
AutomationUtils.NOTE_UUID	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*

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

At Pre	
AutomationUtils.NOTE_TEXT	*No*
AutomationUtils.NOTE_UUID	*No*

Delete Note

At Pre	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_UUID	*No*
At post	
Variable	***For changing***
AutomationUtils.NODE_BASE	*No*
AutomationUtils.NOTE_UUID	*No*

General

Text extraction

At Pre	
Variable	***For changing***
AutomationUtils.DOCUMENT_NODE	*No*

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 in .csv format containing all the Exchange Legacy DNs and their corresponding email addresses. With this data, the database can be altered so the Exchange Legacy DN 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);
                    }
            }
```

```
}

# 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 a document with a 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 features. That will lock UI access to the user.*

Available parameters:

- *The parameter **uuid** is the document UUID.*
- *The parameter **inline** sets whether 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 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 documents 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 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 email** that will be set.*

*The string "**newRealName**" is the **user's 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 emails. For example, create, move, or delete emails.*
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 or 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 statistics for the repository.*
OKMUserConfig	*Manage user home configuration.*

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

OKMActivity

With OKMActivity you can explore the OpenKM activity log data.

Methods

findByFilter

Description:

Method	Return values	Description
findByFilter(String token, ActivityFilter filter)	**List<Activity>**	*Returns a list of all 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

Description:

Method	*Return values*	*Description*
countByFilter(String token, ActivityFilter filter)	***int***	*Returns the number of activity log rows.*

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.api.OKMActivity;
import com.openkm.db.bean.ActivityFilter;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMActivity okmActivity = ContextWrapper.getContext().getBean(OKMActivity.class);

            Calendar beginDate = Calendar.getInstance();
            beginDate.add(Calendar.MONTH, -1);
            Calendar endDate = Calendar.getInstance();
            String item = "f84a2e1f-a858-4e53-9c09-36519d903782";
            String user = "okmAdmin";

            ActivityFilter filter = new ActivityFilter();
            filter.setBegin(beginDate);
            filter.setEnd(endDate);
            filter.setItem(item);
            filter.setUser(user);
            filter.setPage(0);
            filter.setLength(20);
            System.out.println(okmActivity.countByFilter(null, filter));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActions

Description:

Method	*Return values*	*Description*
getActions(String token)	***List<String>***	*Returns a list of all 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();
        }
    }
}
```

OKMAuth

Used for managing security and users. For example, add or remove grants on a node, create or modify users, or get profiles.

Basics

*The class **com.openkm.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or retrieve security grants.*

To set READ and WRITE access, do the following:

```
int permission = Permission.READ + Permission.WRITE;
```

To check if you have permission, do the following:

```
// permission is a valid integer value
if ((permission | Permission.WRITE) = Permission.WRITE) {
  // Has WRITE grants.
}
```

*In almost all methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid node **UUID** (folder, document, mail, record).*

Example of 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, 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 should use the "**user session**".*

*In special cases you might be "**promoted to Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

login

Description:

Method	Return values	Description

login()	***void***	*Simulates the user UI login process.*

When a user is logged in from the UI, some background processes are executed that create 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 may not be created and will raise an error, so it is necessary to execute the login method at the beginning.

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.*

The login token, by default, has 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 by a newer one.

When a user logs into the application for the first time, an internal method is called 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 via the API to

create the specific user folder structure.

Otherwise you can get errors, for example PathNotExistsException /okm:trash/userId when deleting objects, because the folders have not been created.

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.login("userId","password");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantUser

Description:

Method	*Return values*	*Description*
grantUser(String token, String nodeId, String user, int permissions, boolean recursive)	***void***	*Add a user grant on a node.*
The parameter 'recursive' only applies 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, t

            // 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 a user grant on a node.*
The parameter 'recursive' only applies 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);

            // 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();
        }
    }
}
```

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 applies 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 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 a role grant on a node.*
The parameter 'recursive' only applies 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);
            // Remove ROLE_USER write grants at the node but not descendants
            okmAuth.revokeRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_USER", Permission.ALL_GR

            // Remove all ROLE_ADMIN grants to the node and descendants
            okmAuth.revokeRole(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "ROLE_ADMIN", Permission.ALL_G
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGrantedRoles

Description:

Method	*Return values*	*Description*
getGrantedRoles(String token, String nodeId)	***Map<String, Integer>***	*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<CommonUser>***	*Returns the list of all users.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (CommonUser 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<CommonUser>***	*Returns the list of all users who have been assigned a role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
```

```
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            for (CommonUser 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

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();
        }
    }
}
```

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();
        }
    }
}
```

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

createRole

Description:

Method	*Return values*	*Description*
createRole(String token, Role role)	***void***	*Create a new role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.Role;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            Role role = new Role();
            role.setId("ROLE_TEST");
            role.setActive(true);
            okmAuth.createRole(null, role);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRole

Description:

Method	*Return values*	*Description*
deleteRole(String token, String role)	***void***	*Delete a role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.deleteRole(null, "ROLE_TEST");
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

updateRole

Description:

Method	*Return values*	*Description*
updateRole(String token, String role, boolean active)	***void***	*Update a role.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;

public class Test {
    public static void main(String[] args) {
        try {
            OKMAuth.getInstance().updateRole(null, "ROLE_TEST",true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

assignRole

Description:

Method	*Return values*	*Description*
assignRole(String token, String user, String role)	***void***	*Assign a role to a user.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.assignRole(null, "test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

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

```
}
```

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.*

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

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*	*Execute the logout method on the OpenKM side.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            okmAuth.logout(null);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getUser

Description:

Method	*Return values*	*Description*
getUser(String token, String userId)	***CommonUser***	*Returns user data.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.bean.CommonUser;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
            CommonUser 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 applies when the UUID is a folder or record node. *When the parameter 'recursive' is true, the changes 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);
            // 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 applies when the UUID is a folder or record node. *When the parameter 'recursive' is true, the changes 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);
            // Update permissions of ROLE_USER at the node but not descendants
            okmAuth.setRolePermissions(null, "3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_USER", Permission

            // 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 can have only one assigned.*

Example:

```
package com.openkm;

import com.openkm.api.OKMAuth;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

            long tenantId = 1;
            okmAuth.setUserTenant(null, tenantId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isPasswordExpired

Description:

Method	*Return values*	*Description*
isPasswordExpired(String token)	*Boolean*	*Check whether 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 values*	*Description*
resetPassword(String	***void***	*Sends an email with a link that, if the user clicks it, will reset the*

userId)		*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();
        }
    }
}
```

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

OKMBookmark

Used for managing users' bookmarks. For example, add or remove a bookmark on a node for a user.

Basics

*On almost all methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid node **UUID** (folder, document, mail, record).*

Example of nodeId:

- *Using UUID -> "**3887ae75-dd67-4019-b4d1-2f6962815403**";*

*Also, 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 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

add

Description:

Method	Return values	Description
add(String token, String nodeId, String name)	**Bookmark**	Add a new bookmark to 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 {
```

```
    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***	*Rename a bookmark.*
The parameter bmId is the value of the unique bookmark identifier.		

Example:

```
package com.openkm;

import com.openkm.api.OKMBookmark;
import com.openkm.db.bean.Bookmark;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMBookmark okmBookmark = ContextWrapper.getContext().getBean(OKMBookmark.class);
            int bmId = 2;
            Bookmark bookmark = okmBookmark.rename(null, bmId, "newname.doc");
            System.out.println(bookmark);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAll

Description:

Method	*Return values*	*Description*

getAll(String token)	***List<Bookmark>***	*Gets all the bookmarks of the logged-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();
        }
    }
}
```

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 need to 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

getUserCheckedOutDocuments

Description:

Method	*Return values*	*Description*
getUserCheckedOutDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents checked out by the user.*

- *Since OpenKM 7.1.23, pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26, the method returns a List of DashboardDocumentResult elements.*

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 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;
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.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult elements.*

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.*

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

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.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLockedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedRecords

Description:

Method	*Return values*	*Description*
getUserLockedRecords(String token, long offset,	***DashboardRecordResultSet***	*Get the list of records locked by*

long limit)		*the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardRecordResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLockedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedFolders

Description:

Method	*Return values*	*Description*
getUserLockedFolders(String token, int offset, int limit)	***DashboardFolderResultSet***	*Returns a list of folders locked by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.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.getUserLockedFolders(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

getUserLockedMails

Description:

Method	*Return values*	*Description*
getUserLockedMails(int offset, int limit)	***DashboardResultSet***	*Returns a list of emails locked by the user.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardResultSet resultSet = okmDashboard.getUserLockedMails(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedDocuments

Description:

Method	*Return values*	*Description*
getUserSubscribedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents the user is subscribed to.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserSubscribedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedFolders

Description:

Method	*Return values*	*Description*
getUserSubscribedFolders(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of folders the user is subscribed to.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardFolderResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserSubscribedFolders(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedRecords

Description:

Method	*Return values*	*Description*
getUserSubscribedRecords(String token, long offset, long limit)	***DashboardRecordResultSet***	*Get the list of records the user is subscribed to.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardRecordResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserSubscribedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedDocuments

Description:

Method	*Return values*	*Description*
getUserLastCreatedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents most recently created by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult elements.*

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.*

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

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.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult*

elements.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastCreatedDocumentsNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedFolders

Description:

Method	Return values	Description
getUserLastCreatedFolders(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of folders most recently created by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardFolderResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserLastCreatedFolders(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedFoldersNotes

Description:

Method	*Return values*	*Description*
getUserLastCreatedFoldersNotes(String token, long offset, long limit)	***DashboardFolderResultSet***	*Get the list of folders/notes most recently created by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardFolderResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.bean.DashboardFolderResultSet;
import com.openkm.util.ContextWrapper;
```

```
public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardFolderResultSet resultSet = okmDashboard.getUserLastCreatedFoldersNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardFolderResult dashboardFolderResult : resultSet.getResults()) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedRecords

Description:

Method	*Return values*	*Description*
getUserLastCreatedRecords(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of records most recently created by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardRecordResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLastCreatedRecords(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedRecordsNotes

Description:

Method	*Return values*	*Description*
getUserLastCreatedRecordsNotes(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of records/notes most recently created by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardRecordResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*

- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.bean.DashboardRecordResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardRecordResultSet resultSet = okmDashboard.getUserLastCreatedRecordsNotes(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardRecordResult dashboardRecordResult : resultSet.getResults()) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastDownloadedDocuments

Description:

Method	*Return values*	*Description*
getUserLastDownloadedDocuments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of documents last downloaded by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardDocumentResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*

- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastDownloadedDocuments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMails

Description:

Method	*Return values*	*Description*
getUserLastImportedMails(String token, long offset, long limit)	***DashboardMailResultSet***	*Get the list of mails most recently imported by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*

- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardMailResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardMailResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardMailResult dashboardMailResult : okmDashboard.getUserLastImportedMails(null, "okmAdmi
                System.out.println(dashboardMailResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedMailsNotes

Description:

Method	*Return values*	*Description*

getUserLastCreatedMailsNotes(String token, long offset, long limit)	***DashboardMailResultSet***	*Get the list of mails/notes most recently imported by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardMailResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardMailResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardMailResult dashboardMailResult : okmDashboard.getUserLastCreatedMailsNotes(null, "okm
                System.out.println(dashboardMailResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMailAttachments

Description:

Method	*Return values*	*Description*
getUserLastImportedMailAttachments(String token, long offset, long limit)	***DashboardDocumentResultSet***	*Get the list of mail attachments most recently imported by the user.*

- *Since OpenKM 7.1.23 version pagination was added with offset and limit parameters.*
- *In versions of OpenKM older than 7.1.26 the method return a List of DashboardMailResult elements.*

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- *The parameter "limit" is used to limit the number of results returned.*
- *The parameter "offset" says to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;
import com.openkm.ws.rest.util.DashboardDocumentResultSet;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            DashboardDocumentResultSet resultSet = okmDashboard.getUserLastImportedMailAttachments(null, 0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardDocumentResult dashboardDocumentResult : resultSet.getResults()) {
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserDocumentsSize

Description:

Method	*Return values*	*Description*
getUserDocumentsSize(String token)	***long***	*Return 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 history 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" items saved by the user.*

The "user saved" searches are executed periodically to retrieve 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" search.*
The parameter 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 users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 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 users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 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 users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastWeekTopModifiedDocu
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastMonthTopModifiedDocuments

Description:

Method	*Return values*	*Description*
getLastMonthTopModifiedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the top modified documents during the last month by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastMonthTopModifiedDoc
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastModifiedDocuments

Description:

Method	*Return values*	*Description*
getLastModifiedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the last modified documents by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastModifiedDocuments(nu
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedDocuments

Description:

Method	*Return values*	*Description*
getLastCreatedDocuments(String token, long offset, long limit)	***List<DashboardDocumentResult>***	*Get the list of the last created documents by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 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,		*Get the list of the last*

long offset, long limit)	***List<DashboardDocumentResult>***	*documents being edited by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardDocumentResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardDocumentResult dashboardDocumentResult : okmDashboard.getLastCheckedOutDocument
                System.out.println(dashboardDocumentResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedFolders

Description:

Method	Return values	Description
getLastCreatedFolders(String token, long offset, long limit)	***List<DashboardFolderResult>***	*Get the list of the last created folders by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardFolderResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardFolderResult dashboardFolderResult : okmDashboard.getLastCreatedFolders(null, 0, 5)) {
                System.out.println(dashboardFolderResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLastCreatedRecords

Description:

Method	*Return values*	*Description*
getLastCreatedRecords(String token, long offset, long limit)	***List<DashboardRecordResult>***	*Get the list of the last created records by all users.*

Since OpenKM 7.1.23, pagination was added with offset and limit parameters.

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 com.openkm.api.OKMDashboard;
import com.openkm.bean.DashboardRecordResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDashboard okmDashboard = ContextWrapper.getContext().getBean(OKMDashboard.class);
            for (DashboardRecordResult dashboardRecordResult : okmDashboard.getLastCreatedRecords(null, 0, 5)) {
                System.out.println(dashboardRecordResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

visitNode

Description:

Method	*Return values*	*Description*
visitNode(String token, String source, String node, Calendar date)	***void***	*Mark a node as visited.*

From the UI, 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*
- *UserSubscribedRecords*
- *UserLastCreatedDocuments*
- *UserLastCreatedFolders*
- *UserLastCreatedRecords*
- *UserLastImportedMails*
- *UserLastImportedMailAttachments*
- *In case of "user new", the source name is the name of the query.*

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

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 "**null**" value, which indicates the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

create

Description:

Method	*Return values*	*Description*
create(String token, Document doc, InputStream is)	***Document***	*Create a new document.*
The parameter doc must be initialized with the OpenKM path value in the repository. Optionally, you can also set keywords		

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***	*Returns 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 Database query view you can execute a query to verify if 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_renamec
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

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 a document edit.*

This action can only be done by the user who previously executed the checkout action.

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.cancelCheckout(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceCancelCheckout

Description:

Method	*Return values*	*Description*
forceCancelCheckout(String token, String docId)	***void***	*Cancels a document edit.*

This method allows canceling an edit on any document.

It is not mandatory to execute this action as the same user who previously executed the checkout action.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.forceCancelCheckout(null, "f3a977f7-abfc-4f58-a033-988903b31473");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isCheckedOut

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 the 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();
        }
    }
}
```

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 unlocking any locked document.

It is not mandatory to execute this action as 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 java.io.InputStream;
```

```
import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            InputStream is = null;
            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 whether 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.*

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 user's personal trash location - but it is possible to directly purge any document from the whole repository.

> *When a document is purged it can only be restored from a prior 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 values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            okmDocument.move(null, "f3a977f7-abfc-4f58-a033-988903b31473", "b67fe3ae-bcca-4947-b8ea-0cc7aedd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	*Return values*	*Description*
copy(String token, String docId, String dstId)	***void***	*Copy a document into 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 value 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 a 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 total size in bytes of all documents in the document 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 whether the node is 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);
            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 a 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.*

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, 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 and returns a Document object.*
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 document annotations for a specific 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 of a specific 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 documents checked out 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
```

```
            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 which is created by the administrator.*

*The **exprX** is the expression to set the position on the X axis.*

*The **exprY** is the expression to set the position on the Y axis.*

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*

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 which is created by the administrator.*

*The **exprX** is the expression to set the position on the X axis.*

*The **exprY** is the expression to set the position on the Y axis.*

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*

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 using a wizard.*

*The parameter **docPath** should be any valid folder or record **PATH**.*

> *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.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.*

*For 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 the document as 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 any valid folder or record PATH.*		

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMDocument;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            List<String> uuids = new ArrayList<>();
            uuids.add("/okm:root/test/document.pdf");
            uuids.add("/okm:root/test/document1.pdf");
            String fldUuid = "/okm:root/mergue";
            okmDocument.mergePdf(null, fldUuid, "MergePdf.pdf", uuids);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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, 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, Folder fld)	**Folder**	Creates a new folder and returns a Folder object.

*The **path** variable in the **fld** parameter must be initialized. It indicates the folder path in OpenKM.*

```
Folder fld = new Folder();
fld.setPath("/okm:root/test");
```

Other variables of the Folder (fld) will not have any effect on folder creation.

We suggest using the method below to create the folder, rather than this one.

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
            Folder fld = new Folder();
            fld.setPath("/okm:root/test");
            okmFolder.create(null, fld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createSimple

Description:

Method	*Return values*	*Description*
createSimple(String token, String fldPath)	***Folder***	*Creates a new folder and returns 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.*

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 user trash locations - but it is possible to directly purge any folder from the entire repository.

> *When a folder is purged, it will only be able to be restored from a previous repository backup. The purge action permanently removes the folder 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:

```
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();
        }
    }
}
```

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

extendedCopy

Description:

Method	*Return values*	*Description*
extendedCopy(String token, String fldId, String dstId, ExtendedAttributes extAttr)	***void***	*Copies a folder with the associated data into some folder or record.*

The values of the dstId parameter should be a folder or record UUID.

By default, only the binary data and the security grants 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);
```

```
        } 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:

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 a docId:

- *Using UUID -> "**c41f9ea0-0d6c-45da-bae4-d72b66f42d0f**"*

*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

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*
createFolder(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:

```
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, OS, etc. is also using the hardware.*

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMImport;
import com.openkm.db.bean.NodeFolder;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMImport okmImport = ContextWrapper.getContext().getBean(OKMImport.class);
            File fld = new File("/home/openkm/import/");
            okmImport.importFolder(null, "0a19e616-1d11-4ef7-a7c7-f1e109f8ea3c", fld, 4);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMMail

Basics

*In almost all methods you'll see a parameter named "**mailId**". The value of this parameter can be any valid mail **UUID**.*

Example of mailId:

- *Using UUID -> "**0747c0cb-5e4a-4088-92a9-ab97b401af75**";*

*About the parameter named "**fldId**", the value of this parameter can be any 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 Cron tasks are executed in the background without authentication, the methods used in this scenario might use the token parameter. From the default application execution context you must use "**null**" value, which indicates 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:

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 can only 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 a 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 the 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 its name.

> *Only the security grants are copied to the 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 parameter newName value is null, mail will preservate the same name.

By default only the binary data and the security grants, the metadata, keywords, etc. of the mail are not

copyed.

Additional extended attributes parameters:

- *When the category parameter is true the original values of the categories will be copied.*
- *When the keywords parameter is true the original values of the keywords will be copied.*
- *When the propertyGroups parameter is true the original values of the metadata groups will be copied.*
- *When the notes parameter is true the original values of the notes will be copied.*
- *When the wiki parameter is true the original values of the wiki will be copied.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.ExtendedAttributes;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            ExtendedAttributes extAttr = new ExtendedAttributes();
            extAttr.setNotes(true);
            extAttr.setKeywords(true);
            extAttr.setCategories(true);
            extAttr.setPropertyGroups(true);
            okmMail.extendedCopy(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75", "3636f3bb-779c-4851-b145-0d53
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	*Return values*	*Description*
getChildren(String token, String fldId)	***List<Mail>***	*Returns a list of all mails whose parent is mailId.*
The parameter mailId can be a folder or a record node.		

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            for (Mail mail : okmMail.getChildren(null, "3636f3bb-779c-4851-b145-0d53e0702b0f")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	*Return values*	*Description*
isValid(String token, String mailId)	***boolean***	*Returns a boolean that indicates if the node is a mail or not.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            boolean valid = okmMail.isValid(null, "0747c0cb-5e4a-4088-92a9-ab97b401af75");
            System.out.println(valid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	*Return values*	*Description*
getPath(String token, String uuid)	***String***	*Converts a folder UUID to a mail path.*

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

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 an 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 are a list of destination email addresses.*
- *subject is the mail subject.*
- *ccRecipients and bccRecipients are lists of destination email addresses (optional)*
- *replyToMails are a list of email addresses for replies (optional)*
- *docsId are 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();
        }
    }
}
```

sendMail

Description:

Method	*Return values*	*Description*
sendMail(String token, String from, List<String> recipients, String subject, String body)	***void***	*Sends a mail with a specified sender address.*

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");
            String from = "sender@example.com";
            okmMail.sendMail(null, from, recipients, "Testing sending mail from OpenKM", body.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMailWithAttachments

Description:

Method	*Return values*	*Description*
sendMailWithAttachments(String token, String from, 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 an attachment and a specified sender address.*

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:

- *from is the sender email address.*
- *toRecipients are a list of destination email addresses.*
- *subject is the mail subject.*
- *ccRecipients and bccRecipients are lists of destination email addresses (optional)*
- *replyToMails are a list of email addresses for replies (optional)*
- *docsId are 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
            String from = "sender@example.com";

            Mail mail = okmMail.sendMailWithAttachments(null, from, toRecipients, ccRecipients, bccRecipients, replyTo
                    "some subject", "some body", docsId, "b67fe3ae-bcca-4947-b8ea-0cc7aedd4474");
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

*MethodReturn 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***	*Retrieve mail content (binary data).*

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            OutputStream fos = new FileOutputStream("/home/openkm/test.eml");
```

```
            InputStream is = okmMail.getContent(null, "b405a504-d8cb-4166-ac51-22f68acee8c5");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

Method	*Return values*	*Description*
createWizard(String token, String path, String title, InputStream is, String type)	***WizardNode***	*Create a new email with a wizard.*

*The parameters **path** should be any valid folder or record **PATH**.*

Available types:

- *Mail.ORIGIN_MSG*
- *Mail.ORIGIN_EML*

The WizardNode contains a list of pending actions 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 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, through the API only administrators can perform database queries to retrieve this information. Check Repository samples for accessing database level.*

> *More information at Statistics.*

Example:

```
package com.openkm;

import com.openkm.api.OKMMail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMMail okmMail = ContextWrapper.getContext().getBean(OKMMail.class);
            String text = okmMail.getExtractedText(null, "46762f90-82c6-4886-8d21-ad3017dd78a7");
            System.out.println(text);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

forwardEmail

Description:

Method	*Return values*	*Description*
forwardEmail(String token, String uuid, List<String> users, List<String> roles, List<String> mails, String message)	***void***	*Forward an email to a list of users, roles, or external 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**";*

*For 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**";*

*For 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 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:

```
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

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();
        }
    }
}
```

getChildrenPaginated

Description:

Method	*Return values*	*Description*
getChildrenPaginated(String token, String nodeId, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get children nodes paginated.*

*The 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 returning results.*

*The parameter "**orderByFilter**" can have the following values: name or created.*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*

- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.PageInfo;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<Class<? extends NodeBase>> filteredByNodeTypeList = new ArrayList<>();
            filteredByNodeTypeList.add(NodeDocument.class);
            filteredByNodeTypeList.add(NodeFolder.class);
            filteredByNodeTypeList.add(NodeMail.class);
            filteredByNodeTypeList.add(NodeRecord.class);

            // nodeId Folder and Record UUID or Path
            String nodeId = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            String orderByField = "created";
            boolean orderAsc = true;
            PageInfo pageInfo = okmNode.getChildrenPaginated(null, nodeId, 0, 10, "", orderByField, orderAsc,
                    filteredByNodeTypeList);
            System.out.println("Filtered Elements: " + pageInfo.getFilteredElements());
            System.out.println("Total Elements: " + pageInfo.getTotalElements());
            for (Object object : pageInfo.getObjects()) {
                NodeBase nodeBase = (NodeBase) object;
                System.out.println(nodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenByCategoryPaginated

Description:

	Return

Method	values	Description
getChildrenByCategoryPaginated(String token, String nodeId, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Class<? extends NodeBase>> filteredByNodeTypeList)	***PageInfo***	*Get children nodes by category paginated.*

*The 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 returning results.*

*The parameter "**orderByFilter**" can have the following values: name or created.*

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.PageInfo;
import com.openkm.db.bean.NodeBase;
import com.openkm.db.bean.NodeDocument;
import com.openkm.db.bean.NodeFolder;
import com.openkm.db.bean.NodeMail;
import com.openkm.db.bean.NodeRecord;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<Class<? extends NodeBase>> filteredByNodeTypeList = new ArrayList<>();
            filteredByNodeTypeList.add(NodeDocument.class);
```

```
            filteredByNodeTypeList.add(NodeFolder.class);
            filteredByNodeTypeList.add(NodeMail.class);
            filteredByNodeTypeList.add(NodeRecord.class);

            // nodeId Folder and Record UUID or Path
            String nodeId = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            String orderByField = "created";
            boolean orderAsc = true;
            PageInfo pageInfo = okmNode.getChildrenByCategoryPaginated(null, nodeId, 0, 10, "", orderByField, orderA
                    filteredByNodeTypeList);
            System.out.println("Filtered Elements: " + pageInfo.getFilteredElements());
            System.out.println("Total Elements: " + pageInfo.getTotalElements());
            for (Object object : pageInfo.getObjects()) {
                NodeBase nodeBase = (NodeBase) object;
                System.out.println(nodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getBreadcrumb

Description:

Method	*Return values*	*Description*
getBreadcrumb(String token, String fldId)	***List<BreadCrumbItem>***	*Get breadcrumb.*

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.api.OKMNode;
import com.openkm.bean.BreadCrumbItem;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            List<BreadCrumbItem> list = okmNode.getBreadcrumb(null, "39479efe-de5e-468e-91a7-24d2aa3f8837");
            for (BreadCrumbItem item : list) {
                System.out.println(item.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

subscribe

Description:

Method	Return values	Description
subscribe(String token, String nodeId)	**void**	*Adds a subscription to a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.subscribe(null, "39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unsubscribe

Description:

Method	Return values	Description
unsubscribe(String token, String nodeId)	**void**	*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

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**	*Unzips 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***	*Evaluates 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 origin. When possible, the application will try to restore it to the same origin (sometimes this is not possible because the origin no longer exists or security has changed in the origin, etc.).*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.bean.Node;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            Node node = okmNode.restore(null, "0f6463f3-4d36-4091-b518-4fe7c353ee70");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

hasNodesLockedByOtherUser

Description:

Method	*Return values*	*Description*
hasNodesLockedByOtherUser(String token, String nodeId)	***boolean***	*Check if the node and descendants have been locked by other users.*

This method is used to evaluate if it is possible to lock a hierarchy. For example, in the case of a record, the lock affects all the nodes it contains, which will also be locked.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            if (okmNode.hasNodesLockedByOtherUser(null, "0f6463f3-4d36-4091-b518-4fe7c353ee70")) {
                System.out.println("Is blocked");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lock

Description:

Method	*Return values*	*Description*
lock(String token, String nodeId)	***LockInfo***	*Locks a node and returns an object with the Lock information.*

Only the user who locked the document is allowed to unlock it.

A locked document cannot be modified by other users.

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.lock(null, "1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceLock

Description:

Method	*Return values*	*Description*
forceLock(String token, String nodeId)	***void***	*Locks a locked node.*

This method allows locking any 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 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);
            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;
```

```
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 indicating whether the node is locked.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            System.out.println(okmNode.getLockInfo(null, "1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

Method	*Return values*	*Description*
getLockInfo(String token, String nodeId)	***LockInfo***	*Returns an object with the Lock information.*

Example:

```
package com.openkm;
```

```
import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            System.out.println(okmNode.getLockInfo(null, "1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setComment

Description:

Method	*Return values*	*Description*
setComment(String token, String nodeId, String versionName, String comment)	***void***	*Sets the comment for a specific node version.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNode;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNode okmNode = ContextWrapper.getContext().getBean("okmNode", OKMNode.class);
            okmNode.setComment(null, "1ec49da9-1746-4875-ae32-9281d7303a62", "1.14", "Update comment");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

OKMNote

Basics

*On most methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of a nodeId:

- *Using a 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 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

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();
```

```
        }
    }
}
```

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 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 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();
        }
    }
}
```

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

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 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 "**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

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 subscriptors of a node.*

Example:

```
package com.openkm;

import com.openkm.api.OKMNotification;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMNotification okmNotification = ContextWrapper.getContext().getBean(OKMNotification.class);
            for (String subscriptor : okmNotification.getSubscriptors(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338")) {
                System.out.println(subscriptor);
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

notify

Description:

Method	*Return values*	*Description*
notify(String token, String nodeId, List<String> users, List<String> mails, String message, boolean attachment)	***void***	*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();
        }
    }
}
```

OKMProperty

Basics

*On most methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail, or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

*About the parameter named "**catId**", the value of this parameter can be a valid category folder **UUID** node.*

Example of nodeId:

- *Using UUID -> "**5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f**";*

*Also, 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 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:

```
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 for joining).*

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();
        }
    }
}
```

setSigned

Description:

Method	*Return values*	*Description*
setSigned(String token, String nodeId, boolean signed)	***void***	*Marks a document as signed or unsigned 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.*

The parameter nodeId should be a document node.

This method was added in OpenKM version 7.1.28.

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 associate these methods with metadata ones, for historical reasons we continue to maintain the nomenclature.

For more information about Metadata.

*On most methods you'll see a parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID**.*

Example of nodeId:

- *Using UUID -> "**b153c4b7-3d1c-4589-bd42-0ed0f34fd338**";*

The class com.openkm.util.ISO8601 should be used to set and parse metadata date fields. The values of metadata fields of type date are stored in the application in ISO-8601 basic format.

To convert the retrieved metadata field of type date into 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. From the default application execution context you must use the value "**null**", which indicates that the application must use the "**user session**".*

*On special cases you might be "**promoted as Administrator**" using the "**administrator token**".*

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Methods

addGroup

Description:

Method	*Return values*	*Description*
addGroup(String token, String nodeId, String grpName, Map<String, String> properties)	***void***	*Adds 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\"]");
```

*Where \"**t1**\" and \"**t2**\" are valid values and character "**,**" is used as separator.*

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import com.google.gson.Gson;
import com.openkm.Config;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

import java.util.*;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = new HashMap<>();
            Gson gson = new Gson();
            List<String> values = new ArrayList<>();
            values.add("t1");
            values.add("t2");
            properties.put("okp:technology.type", gson.toJson(values));
            properties.put("okp:technology.language", "[ \"java\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.important", String.valueOf(true));
            okmPropertyGroup.addGroup(null, "4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properties)
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeGroup

Description:

Method	*Return values*	*Description*
removeGroup(String token, String nodeId, String	***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();
        }
    }
}
```

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

getAllGroups

Description:

Method	*Return values*	*Description*
getAllGroups(String token)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups defined 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();
        }
    }
}
```

getUserGroups

Description:

Method	*Return values*	*Description*
getUserGroups(String token)	***List<PropertyGroup>***	*Retrieves a list of all metadata groups assigned to 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, String grpName)	***Map<String, String>***	*Retrieves a list of all metadata group elements and their node values.*

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:

```
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:

	Return

Method	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 character "**,**" is used as separator.*

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import com.google.gson.Gson;
import com.openkm.Config;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.util.ISO8601;

import java.util.*;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = new HashMap<>();
            Gson gson = new Gson();
            List<String> values = new ArrayList<>();
            values.add("t1");
            values.add("t2");
            properties.put("okp:technology.type", gson.toJson(values));
            properties.put("okp:technology.language", "[ \"java\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.important", String.valueOf(true));
            okmPropertyGroup.setProperties(null, "4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properti
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupForm

Description:

Method	*Return values*	*Description*
getPropertyGroupForm(String token, String grpName)	***List<FormElement>***	*Retrieves a list of all metadata group 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 that indicates 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();
        }
    }
}
```

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

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 a Metadata Select field type.

More information at Creating your own Suggestion Analyzer and Metadata Select field.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)
            for (String value : okmPropertyGroup.getSuggestions(null, "b153c4b7-3d1c-4589-bd42-0ed0f34fd338", "okg
                    "okp:technology.language")) {
                System.out.println(value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

registerDefinition

Description:

Method	*Return values*	*Description*
registerDefinition(String token, String pgDef, String pgName)	***void***	*Sets the XML metadata groups definition in the repository.*

The method can only be executed by a super user (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 super user (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) with metadata group values assigned to a specific version of a node.*

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.api.OKMPropertyGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)

            Map<String, String> properties = okmPropertyGroup.getProperties(null, "82555dd1-bcc2-4e64-81cb-5f7c2b
            for (String key : properties.keySet()) {
                System.out.println(key + ": " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValue

Description:

Method	*Return values*	*Description*
getSuggestBoxKeyValue(String token, String grpName, String propertyName, String key)	***KeyValue***	*Retrieves an object that contains the key and value of the suggestbox.*

Example:

```
package com.openkm;


import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.KeyValue;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)


            KeyValue kv = okmPropertyGroup.getSuggestBoxKeyValue(null, "okg:groupname", "okp:groupname.proper
            System.out.println(kv.getKey() + ": " + kv.getValue());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

getSuggestBoxKeyValuesFiltered

Description:

Method	*Return values*	*Description*
getSuggestBoxKeyValuesFiltered(String token, String grpName, String propertyName, String filter)	***List<KeyValue>***	*Retrieves a list of objects that contains the key and value of the suggestbox filtered by value.*

Example:

```
package com.openkm;

import java.util.List;
import com.openkm.api.OKMPropertyGroup;
import com.openkm.bean.KeyValue;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMPropertyGroup okmPropertyGroup = ContextWrapper.getContext().getBean(OKMPropertyGroup.class)


            List<KeyValue> kvList = okmPropertyGroup.getSuggestBoxKeyValuesFiltered(null, "okg:groupname", "okp:g

            for (KeyValue kv : kvList) {
                System.out.println(kv.getKey() + ": " + kv.getValue());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

OKMRecord

Basics

*In almost all methods you'll see a parameter named "**recId**". The value of this parameter can be a valid record **UUID**.*

Example of a recId:

- *Using a 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, 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

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 have any effect on 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 a Record object.*

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 purge any record directly from the repository.

> *When a record is purged, it can only be restored from a previous repository backup. The purge action permanently removes the record 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();
        }
```

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

```
    }
}
```

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 another 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 another record.*

The value 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 and its associated data into a folder or another record.*

The value of the dstId parameter should be a folder or record UUID.

When the newName parameter value is null, the record will retain 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***	*Converts a folder UUID to a 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();
        }
    }
}
```

setTitle

Description:

Method	*Return values*	*Description*
setTitle(String token, String recId, String title)	***void***	*Sets the title of 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.setTitle(null, "7f867772-7476-4176-b86f-bdeb8bce1c5f", "new title");
        } 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();
        }
    }
}
```

createMissingRecords

Description:

Method	*Return values*	*Description*
createMissingRecords(String token, String recPath)	***void***	*Creates missing records in the path.*

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.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRecord okmRecord = ContextWrapper.getContext().getBean(OKMRecord.class);
            okmRecord.createMissingRecords(null, "/okm:root/test/test1/test2");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFromTemplate

Description:

Method	*Return values*	*Description*
createFromTemplate(String token, String fldId, String dstPath, Map<String, String> properties, ExtendedAttributes extAttr)	***Record***	*Creates a new record from a template record.*

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.api.OKMRecord;
import com.openkm.bean.ExtendedAttributes;
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);
            Map properties = new HashMap<>();
            ExtendedAttributes extAttr = new ExtendedAttributes();
            Record record = okmRecord.createFromTemplate(null, "7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "/okm:
            System.out.println(record);
        } 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 recPath, String title, long nodeClass)	***WizardNode***	*Creates a new record using a wizard and node class.*

*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();
        }
    }
}
```

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

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 "**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

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 values:

- *NodeRelationType.BIDIRECTIONAL*
- *NodeRelationType.PARENT_CHILD*
- *NodeRelationType.MANY_TO_MANY*

More information at Relation types.

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationType;
import com.openkm.db.bean.NodeRelationType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationType rt : okmRelation.getRelationTypes(null, NodeRelationType.BIDIRECTIONAL)) {
                System.out.println(rt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

add

Description:

Method	*Return values*	*Description*
add(String token, String nodeAId, String nodeBId, long relationType)	***void***	*Sets a relation between two nodes.*
The parameters ***nodeAId*** *and* ***nodeBId*** *should be valid document, folder, mail, or record* ***UUIDs****.*		

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 be deleted only 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.*

For a relation group it only makes sense to apply a relation type of NodeRelationType.MANY_TO_MANY.

> *More information at Relation types.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationType;
import com.openkm.db.bean.NodeRelationType;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationType rt : okmRelation.getRelationTypes(null, NodeRelationType.MANY_TO_MANY)) {
                if (rt.getTitle().equals("staple")) {
                    okmRelation.addGroup(null, "d168924d-e801-426d-a222-00124a1461bd", "staple group", rt.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addNodeToGroup

Description:

Method	*Return values*	*Description*
addNodeToGroup(String token, String nodeId, long groupId)	***void***	*Adds a node to an existing relation group.*

For a relation group it only makes sense to apply a relation type of NodeRelationType.MANY_TO_MANY.

> *More information at Relation types.*

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            for (RelationGroup rg : okmRelation.getRelationGroups(null, "d168924d-e801-426d-a222-00124a1461bd")) {
                if (rg.getRelationTitle().equals("staple")) {
                    okmRelation.deleteGroup(null, "d168924d-e801-426d-a222-00124a1461bd", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteGroup

Description:

Method	*Return values*	*Description*
deleteGroup(String token, String nodeId, long groupId)	***void***	*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

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();
        }
    }
}
```

getRelationGroups

Description:

Method	*Return values*	*Description*
getRelationGroups(String token, long relationTypeId, String filter, int offset, int limit)	***RelationGroupResultSet***	*Retrieves a paginated list of relation groups filtered by relation type and name.*

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*

The parameter "filter" is used to search for relation groups whose name contains the given text.

The parameter "relationTypeId" specifies which type of relation to filter by.

Example:

```
package com.openkm;

import com.openkm.api.OKMRelation;
import com.openkm.bean.RelationGroup;
import com.openkm.util.ContextWrapper;
import com.openkm.bean.RelationGroupResultSet;

public class Test {

    public static void main(String[] args) {
        try {
            OKMRelation okmRelation = ContextWrapper.getContext().getBean(OKMRelation.class);
            // Get relation groups of type 1, filtering by name containing "invoice", with pagination
```

```
            long relationTypeId = 1;
            String filter = "invoice";
            int offset = 0;
            int limit = 10;
            RelationGroupResultSet resultSet = okmRelation.getRelationGroups(null, relationTypeId, filter, offset, limit);
            System.out.println("Total results: " + resultSet.getTotal());
            for (RelationGroup group : resultSet.getResults()) {
                System.out.println(group);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMReport

Basics

You can set several report formats based on the 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 variable **active** is set to **true**, only the active reports will be returned; otherwise, all reports will be 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) {
```

```
            e.printStackTrace();
        }
    }
}
```

get

Description:

Method	*Return values*	*Description*
get(String token, long rpId)	***Report***	*Returns a Report object.*
*The variable **rpId** is a 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, long rpId, Map<String, Object> params, String format)	***InputStream***	*Returns the InputStream of the report.*
*The variable **rpId** is a 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

*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

getRootFolder

Description:

Method	*Return values*	*Description*
getRootFolder(String token)	***Folder***	*Returns the Folder object for the 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 the 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 returned folder 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 the 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 the 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 the 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 returned folder 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 the 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 the node "/okm:mail/{userId}".*

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

The returned folder will be the user's mail folder.

> *For example, if the method is executed by the user "okmAdmin", then the returned folder 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 the 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();
        }
    }
}
```

getCategoriesFolder

Description:

Method	*Return values*	*Description*
getCategoriesFolder(String token)	***Folder***	*Returns the Folder object for the 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 previous 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 boolean 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();
        }
    }
}
```

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

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);
        } 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 given 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();
        }
    }
}
```

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.* *For searching 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.*
content		*Yes.*	

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

mimeType		*No.*	*The value should be a valid and registered MIME type.* *Only can be applied to documents node.*
language		*No.*	*The value should be a valid language.* *Only can be applied to documents node.*
folder		*No.*	*When empty, 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 applies to mail nodes.*
mailTo		*Yes.*	*Only applies to mail nodes.*
notes		*Yes.*	

properties	***Map<String, String>***	*Yes on almost all.*	*On metadata field values like "date", wildcards cannot be applied.*
			
			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 (
			will be ignored by OpenKM.
			
			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 â€" the marshalling and unmarshalling process â€" while you are only interested in using 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();
        }
    }
}
```

find

Description:

Method	*Return values*	*Description*
find(String token, QueryParams params, String sortField, boolean sortReverse, String propertiesPlugin)	***List<QueryResult>***	*Returns a list of results filtered by the values of the queryParams parameter.*

From version 7.1.27 of OpenKM, this method was added

*The parameter "**propertiesPlugin**" must be the canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {
```

```
    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            for (QueryResult qr : okmSearch.find(null, qParams, "name", true, "com.openkm.plugin.properties.SimpleNo
                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 â€" the marshalling and unmarshalling process â€" while you are only interested in using 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();
        }
    }
}
```

findPaginated

Description:

Method	*Return values*	*Description*
findPaginated(String token, QueryParams params, int offset, int limit, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be the 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 â€" the marshalling and unmarshalling process â€" while you are only interested in using 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, "com.openkm.plugin.properties.SimpleNode
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated

Description:

Method	*Return values*	*Description*
findPaginated(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*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.

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, "name", true, 20, 10);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
  }
```

findPaginated

Description:

Method	*Return values*	*Description*
findPaginated(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be the canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

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

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, "name", true, 20, 10, "com.openkm.plugin.propertie
            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 metadata group name.*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            for (QueryResult qr : okmSearch.findWithMetadata(null, qParams, "com.openkm.plugin.properties.SimpleNo
                    groups)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadata

Description:

Method	*Return values*	*Description*
findWithMetadata(String token, QueryParams params, String sortField, boolean sortReverse, String propertiesPlugin, List<String> groups)	***List<QueryResult>***	*Returns a list of results filtered by the values of the queryParams parameter.*

From version 7.1.27 of OpenKM this method was added

*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.*

*Available **sortField** values:*

```
name
author
lastModified
```

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            for (QueryResult qr : okmSearch.findWithMetadata(null, qParams, "name", true, "com.openkm.plugin.proper
                    groups)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadataPaginated

Description:

Method	*Return values*	*Description*
findWithMetadataPaginated(String token, QueryParams params, int offset, int limit, String propertiesPlugin, List<String> groups)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

*The parameter "**groups**" is the name metadata group.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.db.bean.QueryParams;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            ResultSet rs = okmSearch.findWithMetadataPaginated(null, qParams, 0, 10, "com.openkm.plugin.properties
            System.out.println("Total results:" + rs.getTotal());
```

```
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadataPaginated

Description:

Method	*Return values*	*Description*
findWithMetadataPaginated(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin, List<String> groups)	***ResultSet***	*Returns a list of paginated results filtered by the values of the queryParams parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" used to skip that many results before the beginning to return results.*

*The parameter "**propertiesPlugin**" must be a canonical class name of the class which implements the NodeProperties interface.*

*The parameter "**groups**" is the name metadata group.*

*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.

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*");
            String sortField = "okm:";
            boolean sortReverse = false;

            List<String> groups = new ArrayList<>();
            groups.add("okg:tpl");

            ResultSet rs = okmSearch.findWithMetadataPaginated(null, qParams, sortField, sortReverse, 20, 10, "com.c
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBasePaginated

Description:

Method	*Return values*	*Description*
findSimpleNodeBasePaginated(String token, QueryParams params, String sortField, boolean sortReverse, int offset, int	***SimpleNodeBaseResultSet***	*Returns a list of SimpleNodeBase paginated results filtered by the values of*

limit, String propertiesPlugin)		*the queryParams parameter.*

From version 7.1.19 of OpenKM this method was added

*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 the canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

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();
        }
    }
}
```

findByQuery

Description:

Method	*Return values*	*Description*
findByQuery(String token, String query)	***List<QueryResult>***	*Returns a list of paginated results filtered by the value of the 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 query 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.*

From version 7.1.27 of OpenKM this method was added

*The parameter "**propertiesPlugin**" must be the canonical class name of the class which implements the NodeProperties interface.*

*Available **sortField** values:*

```
name
author
lastModified
```

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (QueryResult qr : okmSearch.findByQuery(null, "text:grial AND name:o*.pdf", "name", true, "com.openkm
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	*Return values*	*Description*
findByQueryPaginated(String token, String query, int offset, int limit)	***ResultSet***	*Returns a list of paginated results filtered by the value of the query parameter.*

*The parameter "**limit**" and "**offset**" allows you to retrieve just a portion of the results of a query.*

- *The parameter "**limit**" is used to limit the number of results returned.*
- *The parameter "**offset**" is used to skip that many results before the beginning to return results.*

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- *limit=10*
- *offset=0*

Now suppose you want to show the results from 11-20, you should use these values:

- *limit=10*
- *offset=10*

*The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:*

- *"name:grial" is filtering field name by word grial.*

More information about lucene sintaxis at Lucene query syntax.

Example:

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.QueryResult;
import com.openkm.bean.ResultSet;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            ResultSet rs = okmSearch.findByQueryPaginated(null, "text:grial AND name:o*.pdf", 10, 10);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	*Return values*	*Description*
findByQueryPaginated(String token, String query, int	***ResultSet***	*Returns a list of paginated results filtered by*

offset, int limit, String propertiesPlugin)		*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 start of the returned 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 - while you are only interested in using a few object variables. If this is your case, you can use NodeProperties classes to retrieve the object variables that 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", 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 start of the returned 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 - while you are only interested in using a few object variables. If this is your case, you can use NodeProperties classes to retrieve the object variables that 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.*

This method was added in OpenKM version 7.1.20.

*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.*

*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 - while you are only interested in using a few object variables. If this is your case, you can use NodeProperties classes to retrieve the object variables that 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.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 queryName variable of the params parameter must 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:

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 saved searches 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 their count values filtered by other keywords.*

Example:

- *Doc1.txt has keywords "test", "one", "two".*
- *Doc2.txt has keywords "test", "one".*
- *Doc3.txt has keywords "test", "three".*

The results filtered by "test" -> "one", "two", "three".

The results filtered by "one" -> "test", "two".

The results filtered by "two" -> "test", "one".

The results filtered by "three" -> "test".

The 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 values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Document doc : okmSearch.getCategorizedDocuments(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f"))
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedFolders

Description:

Method	*Return values*	*Description*
getCategorizedFolders(String token, String categoryId)	***List<Folder>***	*Retrieves a list of all folders related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Folder fld : okmSearch.getCategorizedFolders(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedMails

Description:

Method	*Return values*	*Description*
getCategorizedMails(String token, String categoryId)	***List<Mail>***	*Retrieves a list of all mails related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
```

```
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Mail mail : okmSearch.getCategorizedMails(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedRecords

Description:

Method	*Return values*	*Description*
getCategorizedRecords(String token, String categoryId)	***List<Record>***	*Retrieves a list of all records related to a category.*
The values of the categoryId parameter should be a category folder UUID or path.		

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Record rec : okmSearch.getCategorizedRecords(null, "5cbc35fc-23c0-48f5-a7bc-3cfa1e7be25f")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentsByKeyword

Description:

Method	*Return values*	*Description*
getDocumentsByKeyword(String token, String keyword)	***List<Document>***	*Retrieves a list of all documents related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Document;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Document doc : okmSearch.getDocumentsByKeyword(null, "test")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFoldersByKeyword

Description:

Method	*Return values*	*Description*
getFoldersByKeyword(String token, String keyword)	***List<Folder>***	*Retrieves a list of all folders related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Folder fld : okmSearch.getFoldersByKeyword(null, "test")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailsByKeyword

Description:

Method	*Return values*	*Description*

getMailsByKeyword(String token, String keyword)	***List<Mail>***	*Retrieves a list of all mails related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Mail;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Mail mail : okmSearch.getMailsByKeyword(null, "test")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordsByKeyword

Description:

Method	*Return values*	*Description*
getRecordsByKeyword(String token, String keyword)	***List<Record>***	*Retrieves a list of all records related to a keyword.*

```
package com.openkm;

import com.openkm.api.OKMSearch;
import com.openkm.bean.Record;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMSearch okmSearch = ContextWrapper.getContext().getBean(OKMSearch.class);
            for (Record rec : okmSearch.getRecordsByKeyword(null, "test")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentsByPropertyValue

Description:

Method	Return values	Description
getDocumentsByPropertyValue(String token, String group, String property, String value	***List<Document>***	*Retrieves a list of all documents related 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:

Method	Return 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();
        }
    }
}
```

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

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();
        }
    }
}
```

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*

Sent date 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 11-20, 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 token, String lang, QueryParams params, boolean compact, Ref<String> fileName)	***InputStream***	*Exports a list of results as a CSV filtered by the values of the queryParams parameter.*

Method added in OpenKM version 7.1.27.

*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*");

            Ref fileName = new Ref<>("export.csv");
            InputStream is = ws.search.csvExport(null, "es-ES", params, false, fileName);
            // Or you can use the deprecated method directly with the filename
            OutputStream fos = new FileOutputStream("/home/openkm/okm/export.csv");
```

```
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OKMStats

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 "**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

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:

Method	Return values	Description

getDocumentsByContext(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

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

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.

OKMUserConfig

*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

setHome

Description:

Method	*Return values*	*Description*
setHome(String token, String nodeId)	***void***	*Sets the user's home.*

By default, the user's home value is "/okm:root".

Example:

```
package com.openkm;

import com.openkm.api.OKMUserConfig;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMUserConfig okmUserConfig = ContextWrapper.getContext().getBean(OKMUserConfig.class);
            okmUserConfig.setHome(null, "/okm:root/test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getConfig

Description:

Method	*Return values*	*Description*
getConfig(String token)	***UserConfig***	*Returns the user configuration.*
Returned values are: • *The user ID.* • *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.

Common classes

OpenKM common classes:

OpenKM common classes	*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 logs. For example, to show errors, info, or warnings.*
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 a time.*
GeneralUtils	*Generic utility methods. For example, to notify a user of something.*
ISO8601	*Utility class that provides bidirectional format transformation between Java Time objects and ISO8601 format used for metadata date fields.*
MailUtils	*Utility methods to send mail. 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 an image 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 the actual tenant of the logged-in user.*
StackTraceUtils	*Utility methods for stack traces.*
TemplateUtils	*Utility methods to handle templates. For example, replace text in a template.*

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

LegacySrv	*Utility methods to perform SQL queries 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, to create 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 of the file.*		

Example:

```
package com.openkm;

import java.io.ByteArrayOutputStream;
import java.io.File;

import com.openkm.util.ArchiveUtils;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            ArchiveUtils archiveUtils = ContextWrapper.getContext().getBean(ArchiveUtils.class);
            File file = new File("/home/openkm/test.pdf");
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            archiveUtils.createZip(file, baos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createJar

Description:

Method	*Return values*	*Description*
createJar(File path, String root, OutputStream	***void***	*Recursively create a JAR archive from a*

os)		*directory.*

path*: The file to compress.*

root*: A text specifying the destination of the file.*

os*: The output stream of the file.*

Example:

```
package com.openkm;

import java.io.File;
import java.io.FileOutputStream;
import java.io.StringWriter;

import com.openkm.util.ArchiveUtils;
import com.openkm.util.ContextWrapper;
import com.openkm.util.FileUtils;
import com.openkm.util.PathUtils;
import com.openkm.util.impexp.RepositoryExporter;
import com.openkm.util.impexp.TextInfoDecorator;

public class Test {

    public static void main(String[] args) {
        try {
            ArchiveUtils archiveUtils = ContextWrapper.getContext().getBean(ArchiveUtils.class);
            PathUtils pathUtils = ContextWrapper.getContext().getBean(PathUtils.class);
            FileUtils fileUtils = ContextWrapper.getContext().getBean(FileUtils.class);
            RepositoryExporter repositoryExporter = ContextWrapper.getContext().getBean(RepositoryExporter.class);

            StringWriter out = new StringWriter();

            File dirFile = new File("/home/openkm");
            final File tmpFile = File.createTempFile("test", ".jar", dirFile);
            FileOutputStream os = new FileOutputStream(tmpFile);

            File tmp = fileUtils.createTempDir();
            repositoryExporter.exportFolder(null, "/okm:root/jar", tmp, false, out, new TextInfoDecorator("/okm:root/jar"));

            // Jar files
            archiveUtils.createJar(tmp, pathUtils.getName("/home/openkm"), os);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ExecutionUtils

Execution-related methods. For example, to run a script or JAR.

Methods

runScript

Description:

Method	*Return values*	*Description*
runScript(File script)	***Object[]***	*Execute 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.*
***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 error, info, or warning messages.

Methods

info

Description:

Method	*Return values*	*Description*
info(String baseName, String message, Object... params)	***void***	*Static file logger for info.*
***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");
        } 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");
        } 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");
        } 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 extension.*
***file**: The full name of the file.*		

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

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();
        }
    }
}
```

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

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's 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 the ISO8601 basic format for 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 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 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.*

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:

```
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 an image or text.

Methods

merge

Description:

Method	*Return values*	*Description*
merge(List<InputStream> inputs, OutputStream output)	***void***	*Merge several PDFs into a new one.*
***Inputs**: List of input streams to merge.* ***output**: Output stream for 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);
        }
    }
}
```

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();
        }
    }
}
```

fillForm

Description:

Method	*Return values*	*Description*
fillForm(InputStream input, Map<String, Object> values, OutputStream output)	***void***	*Fill the PDF form with values.*
***input**: The input stream of the PDF file with the form.* ***values:** A map of values to be used to fill out the form.* ***output**: The generated PDF with the form filled in.*		

Example using this file FillForm.pdf:

```
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);
            InputStream fis = new FileInputStream("/home/openkm/FillForm.pdf");
            OutputStream fos = new FileOutputStream("/home/openkm/FillFormOutput.pdf");

            Map<String, Object> values = new HashMap<>();
            values.put("TextField_1", "Field #1 value");
            values.put("TextField_2", "Field #2 value");
            values.put("TextField_3", "Field #3 value");

            pdfUtils.fillForm(fis, values, fos);
        } 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 user ID 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 user ID of the logged-in user.*

The user must be logged in before executing these 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();
        }
    }
}
```

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 the 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) {
```

```
    }
        } catch (Exception e) {
    System.out.println(StackTraceUtilsUtils.toString(e));
    }
   }
  }
```

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

TemplateUtils

Utility methods to handle templates. For example, replace text in a template.

Methods

templateExists

Description:

Method	*Return values*	*Description*
templateExists(String name)	***boolean***	*Checks whether the template exists.*
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 replacement 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.*

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();
        }
    }
}
```

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 only allows a single SQL script.

This action can only be performed by users with administrator privileges.

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, NB
            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 only allows a single HQL script.

This action can only be performed by users with administrator privileges.

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 for the system user.*

> *There is a special super-user named system, executed by OpenKM in the background.*
>
> *The authorization token of the system user is usually used to promote standard user actions or cron jobs to super-user privileges.*

Example:

```
package com.openkm;

import com.openkm.api.OKMDocument;
import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
            String systemToken = DbSessionManager.getInstance().getSystemToken();
            okmDocument.delete(systemToken, "d50133e3-dbfa-4d01-a109-28785cd48f40"); // Delete action is execute
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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

ConfigSrv

getString

Description:

Method	*Return values*	*Description*
getString(String key, String defaultValue)	***String***	*Find the config parameter (as 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***	*Find 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;
```

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

```
public class Test {

    public static void main(String[] args) {
        try {
            ConfigSrv configSrv = ContextWrapper.getContext().getBean(ConfigSrv.class);
            System.out.println("Config parameter: " + configSrv.getText("paramKey", "defaultValue"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getBoolean

Description:

Method	*Return values*	*Description*
getBoolean(String key, String defaultValue)	***boolean***	*Find the config parameter (as 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***	*Find 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***	*Find 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***	*Find 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***	*Find 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:

Method	*Return values*	*Description*
getList(String key, String defaultValue)	***List<String>***	*Find the config parameter (as a list type) by PK with a default value*

Example:

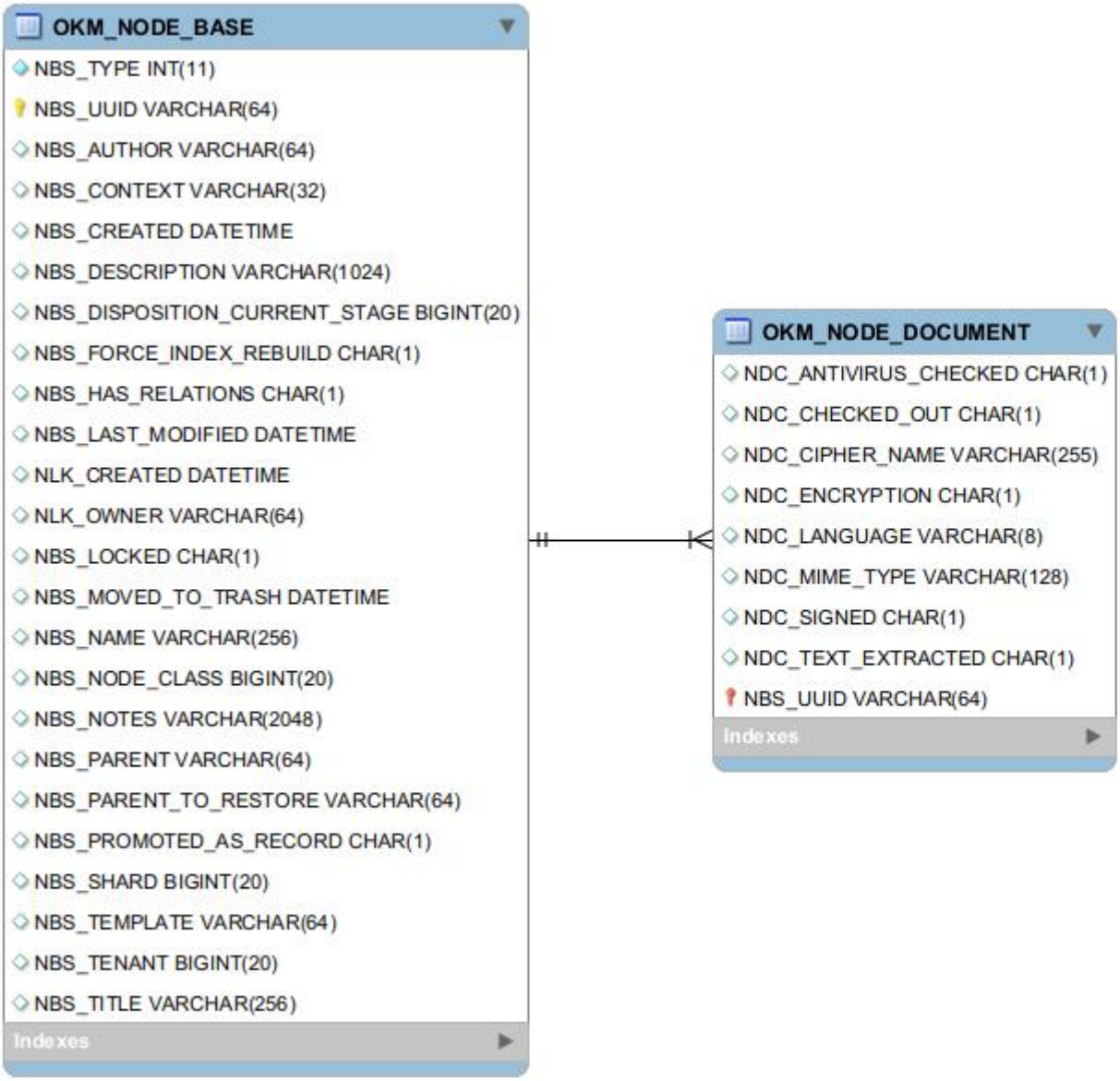
```
package com.openkm;

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 the node types (Document, Folder, Mail, or Record).*

*The field named **NBS_UUID** is the **unique node ID**. **NBS_UUID field** is the **foreign key between the parent table** (OKM_NODE_BASE) **and child tables** (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 emails.*
- ***OKM_NODE_RECORD**: Stores information about records.*

SQL example queries

Query used to retrieve 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 where the name starts with 'test':

```
select * from OKM_NODE_BASE nb
  inner join OKM_NODE_RECORD nr on nr.nbs_uuid = nb.nbs_uuid
  where nb.nbs_name like 'test%'
```

To find a document by UUID, you could use this query:

```
select * from OKM_NODE_BASE nb
  inner join OKM_NODE_DOCUMENT nd on nd.nbs_uuid = nb.nbs_uuid
  where nb.nbs_uuid = 'fc35ce80-57f2-4026-ae77-7a11bca042ed'
```

List of child nodes with standard security:

```
SELECT NB.NBS_UUID FROM OKM_NODE_BASE NB left outer join OKM_NODE_USER_PERMISSION UP ON
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
```

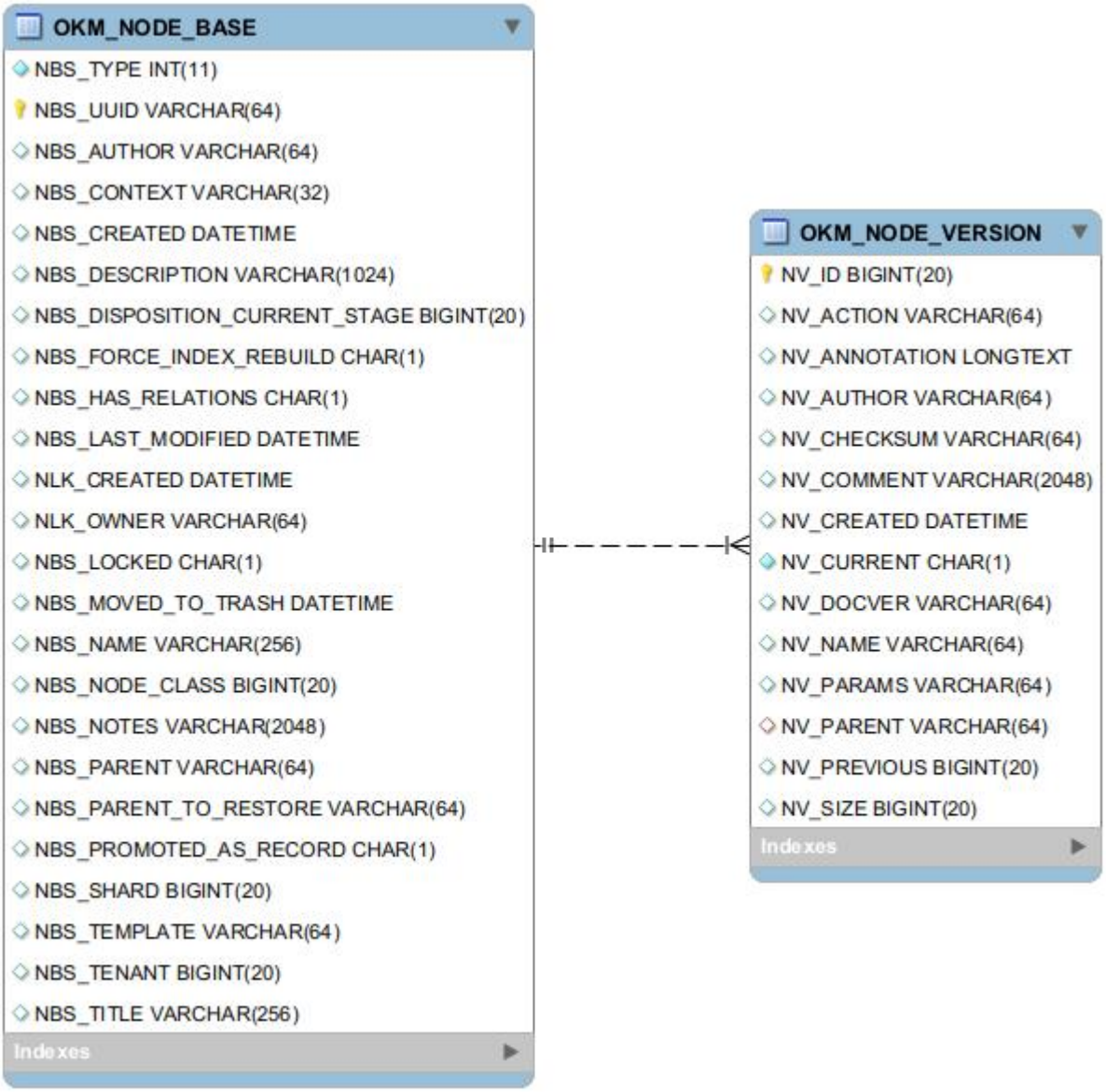
Mark documents to be indexed again

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:

```
update OKM_NODE_DOCUMENT set NDC_TEXT_EXTRACTED='F'
```

Node version

These tables store information about node version history:



*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';
```

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 for a document named 'test.pdf' when its binary was updated:

```
select nv.* from OKM_NODE_DOCUMENT nd
 inner join OKM_NODE_VERSION nv on nv.nv_parent = nd.nbs_uuid
 inner join OKM_NODE_BASE nb on nd.nbs_uuid = nb.nbs_uuid
 where nb.nbs_name = 'test.pdf' and nv.nv_action = 'DOCUMENT_CHANGED';
```

The next query is useful to know where all the versions of a file named 'test.pdf' are stored in the datastore:

```
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'
```

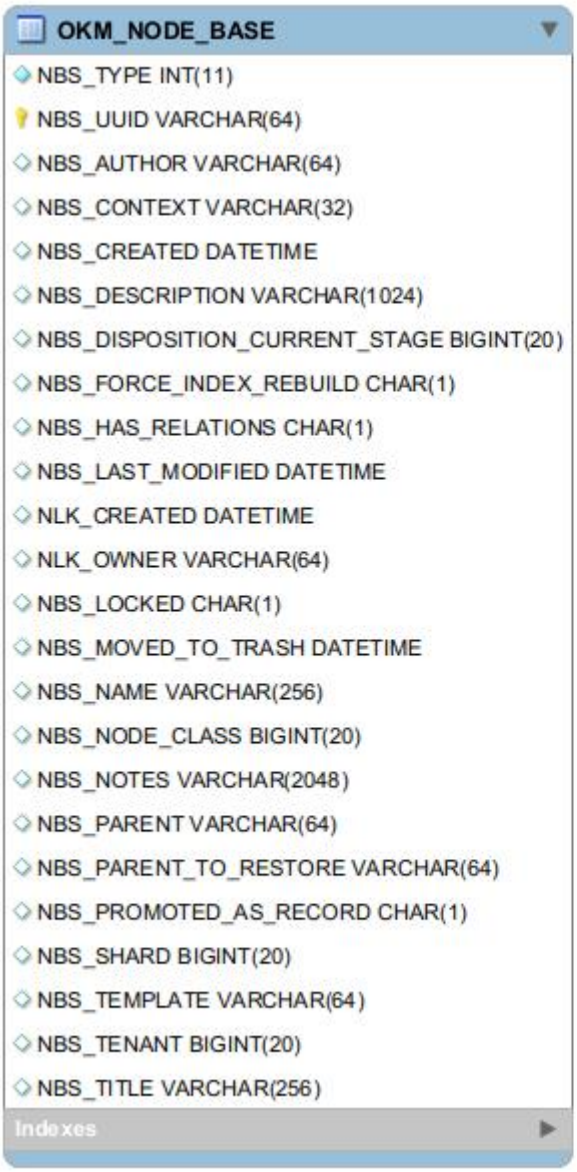
*For example, '**f0d7f0a1**-b337-4917-b592-56e163b81cde'. This means your file is located on the physical drive at*

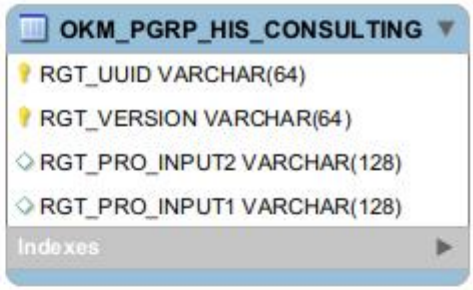
the following path (based on the first eight characters of this ID):

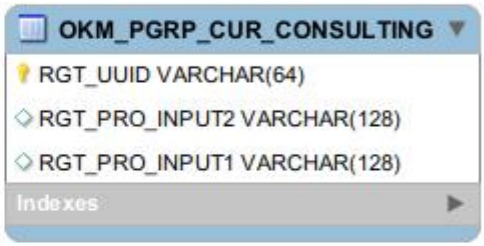
*/home/user/tomcat_home/repository/datastore/**f0/d7/f0/a1**/f0d7f0a1-b337-4917-b592-56e163b81cde*

Node metadata properties

Each metadata group uses two tables to store information; take the following as a sample:







Sample:

Take a look at [the Metadata XML definition](#) for more information.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.10//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.10.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Input label" name="okp:consulting.input1" />
    <separator label="Separator label" name="okp:consulting.separator" />
    <input label="Input label" name="okp:consulting.input2" />
  </property-group>
</property-groups>
```

After registering, the previous metadata definition will create two tables 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 the history of 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*

Query to get all the metadata values from the document named 'test.pdf' with UUID 'e0518d0e-8312-4df9-991c-263f16b061bf' and their changes:

```
SELECT * FROM OKM_PGRP_HIS_CONSULTING where RGT_UUID = 'e0518d0e-8312-4df9-991c-263f16b061bf
```

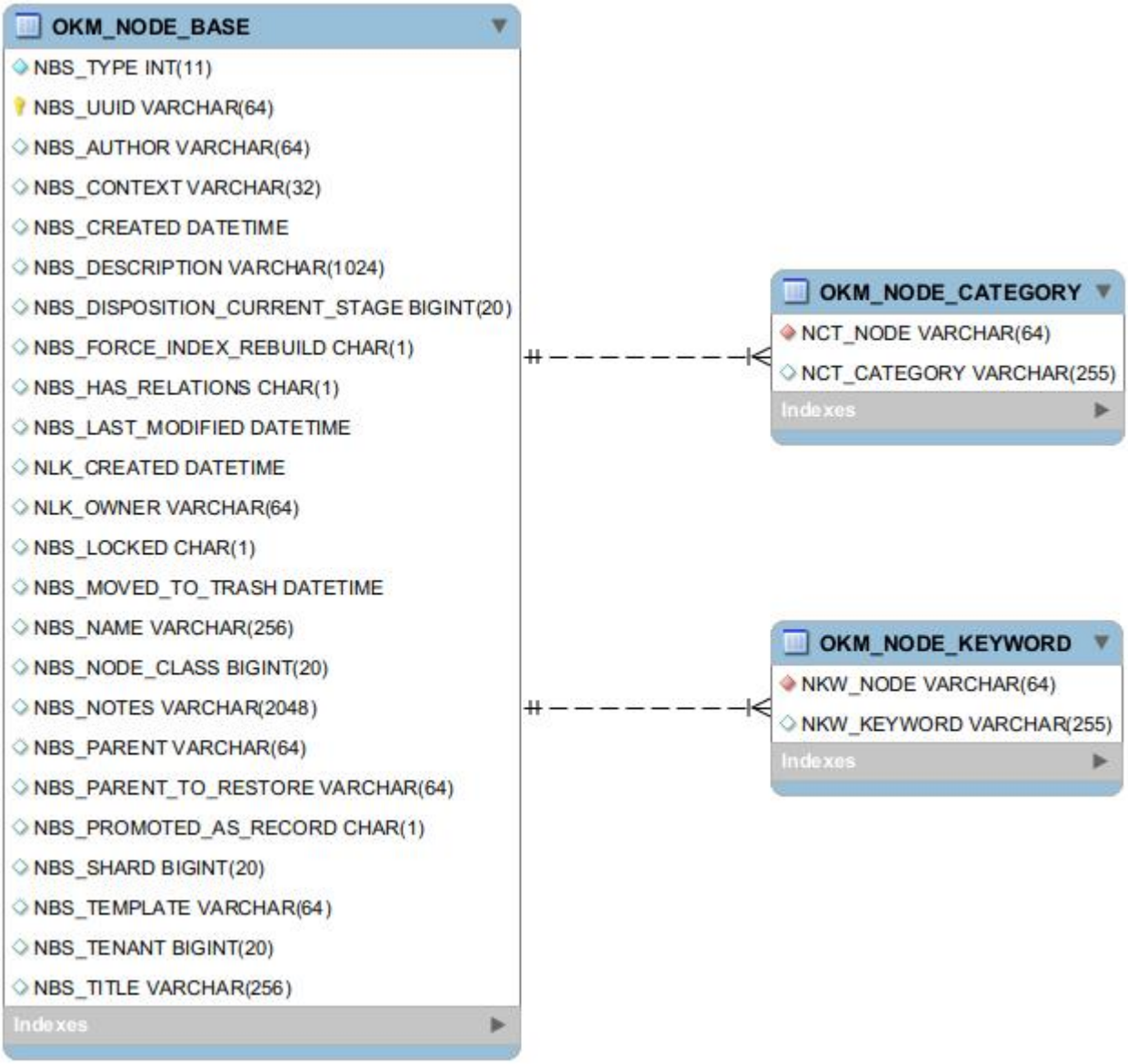
This query will print all metadata values for each version. Usually, the actual version of these metadata values is the one with the largest RGT_VERSION value.

Query to get changes for 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 how to build your user interface based on an external application.

Field / Property	*Description*
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.

Security Provider

Developer information for OpenKM authentication through custom Authentication provider (vía patch) and 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 file called **openkm.properties** 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

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*

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

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 have to 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

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 have 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
 echo "Not able to patch file"
fi
```

Remember to check executable permissions of the prepare.sh script.

Step 3: OpenKM configuration

You need the following 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 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, 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 using KCenter methods

The following sample demonstrates how to access KCenter API methods from an external application with JavaScript.

Step 1 - create a sample HTML file

Create the following sample HTML file and copy it to the webapps/openkm folder with the name ***test.html***

```
<html>
  <head>
    <script type="text/javascript" src="kcenter/api/common.js"></script>
    <script type="text/javascript" src="kcenter/api/browser.js"></script>
  </head>
  <body>
    <h1>Quick sample about how do request to exposed KCenter API</h1>

    <button onclick="_refreshUI()">
      Refresh the UI in the browser view
    </button>
  </body>
</html>
```

Step 2 - Configure the external tab document extension

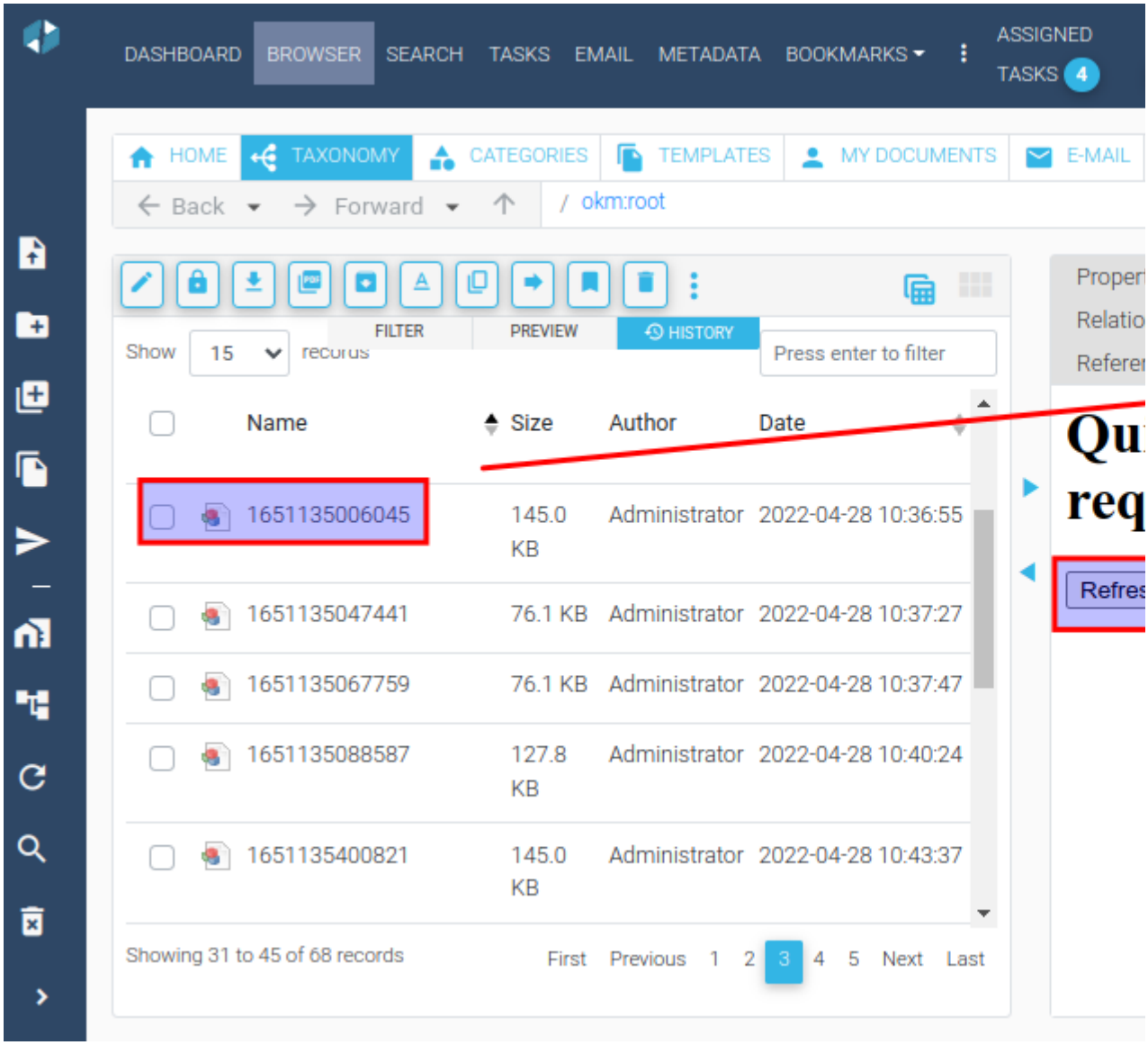
> *More information about the extension at the External tab document extension*
>
> *More information about enabling extensions at Enable extensions (remember to enable the extension in the profile and refresh the KCenter UI).*

- *Configure the external 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

Complete frontend extension sample

This sample shows how to quickly extend OpenKM features with JSP, HTML, CSS and JavaScript technologies using OpenKM Java API and how to use JavaScript functions of the exposed frontend Java methods.

Download the extension sample code complete_extension_sample.zip.

Description

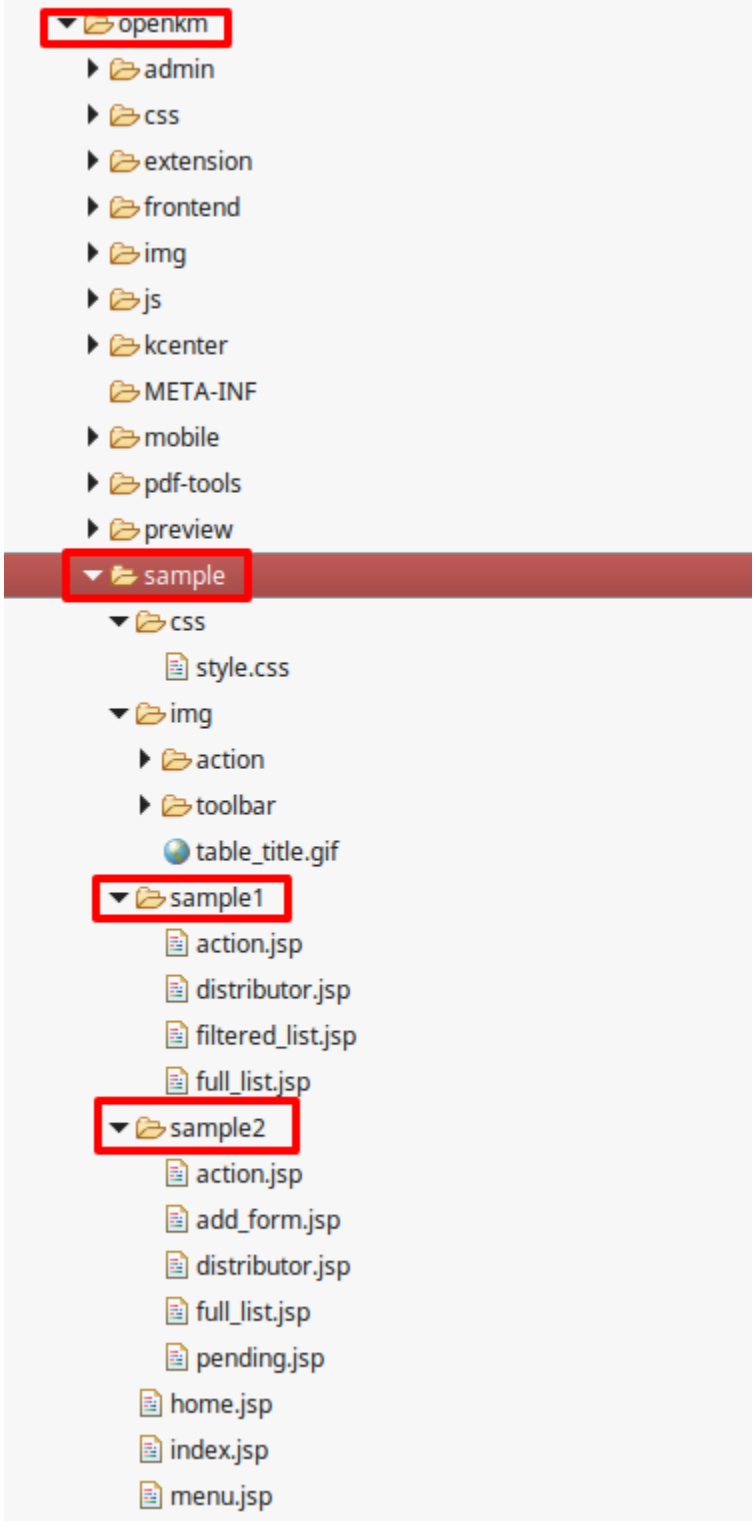
The sample is shown in a new Workspace tab.

The main screen is shown as a toolbar with three buttons:

- *The first button goes to the main screen of the sample.*
- *The second button shows Sample 1.*
- *The third button shows Sample 2.*

Project structure:

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



Installation

- *Stop the application.*
- *Open the contents of the downloaded file into the $TOMCAT_HOME/webapps/openkm folder (or, preferably, into*

openkm.war)

- *Delete $TOMCAT_HOME/work/catalina/localhost*
- *Start the application.*
- *Check the URL http://localhost:8080/openkm/sample.*

*The screenshots were taken in combination with "**ExtraTab**" frontend extension feature; you can also use TabWorkspaceExtension.*

When you change the contents of the openkm.war, the file is deployed as a new version of the application, and all the contents in $TOMCAT_HOME/webapps/openkm will be refreshed, and you will lose any changes you made directly in the webapps/openkm folder.

Implementation details

index.jsp

Index.jsp controls the iframe height with JavaScript.

*Another important point is that, for example, after uploading a document (Sample 2), the parameter **urlToOpen** is used to indicate which URL must be opened.*

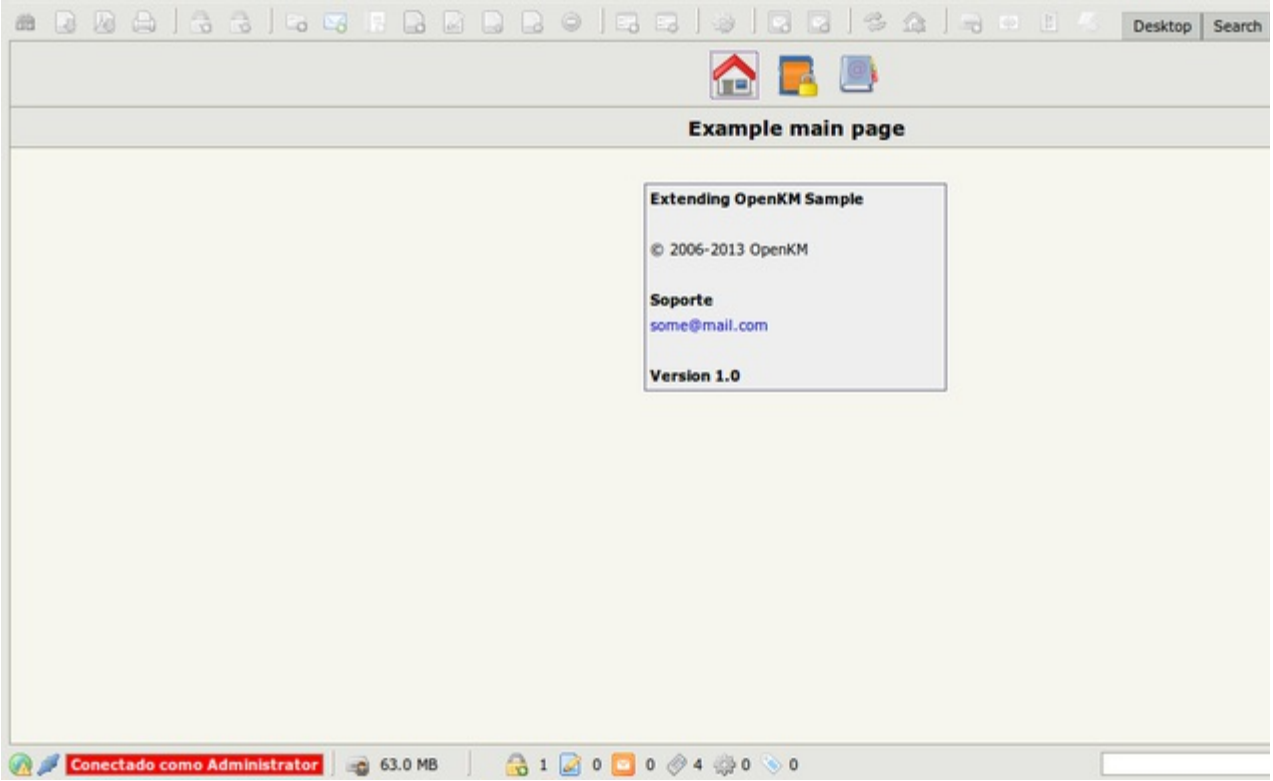
```
HttpSessionManager httpSessionManager = ContextWrapper.getContext().getBean(HttpSessionManager.class);
OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);
WebUtils webUtils = ContextWrapper.getContext().getBean(WebUtils.class);

httpSessionManager.add(request);
okmAuth.login();
String urlToOpen = URLDecoder.decode(webUtils.getString(request, "urlToOpen"), "UTF-8");
if (urlToOpen.equals("")) {
    urlToOpen = "home.jsp";
}
```

menu.jsp

Contains the menu URL definitions.

home.jsp

Sets the main home screen.



Sample 1

Is composed of 4 files:

- *action.jsp (the main controller where the logic is set).*
- *distributor.jsp (the main layer with two columns at 50%).*
- *full_list.jsp (shows the full list with no special JavaScript control).*
- *filtered_list.jsp (shows the filtered list with no special JavaScript control).*

action.jsp

The main controller.

```
<%@page import="java.util.Map"%>
<%@page import="java.util.HashMap"%>
<%@page import="java.util.List"%>
<%@page import="java.util.ArrayList"%>
<%@ page import="java.text.SimpleDateFormat"%>
<%@ page import="java.io.IOException"%>
<%@ page import="java.net.URLDecoder"%>
<%@ page import="org.slf4j.Logger"%>
<%@ page import="org.slf4j.LoggerFactory"%>
<%@ page import="com.openkm.bean.Document"%>
<%@ page import="com.openkm.util.WebUtils"%>
<%@ page import="com.openkm.bean.CommonUser"%>
<%@ page import="com.openkm.api.OKMAuth"%>
<%@ page import="com.openkm.api.OKMRepository"%>
```

```
<%@ page import="com.openkm.api.OKMDocument"%>
<%@ page import="com.openkm.util.ContextWrapper"%>
<%@ page extends="com.openkm.extension.servlet.BaseServlet"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%!
private static Logger log = LoggerFactory.getLogger("com.openkm.sample");
private static SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
private WebUtils webUtils = ContextWrapper.getContext().getBean(WebUtils.class);

public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException,
        ServletException {
    String action = webUtils.getString(request, "action");
    try {
        list(request, response);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        sendErrorRedirect(request, response, e);
    }
}

// Used by fileupload return
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException,
        ServletException {
    String action = webUtils.getString(request, "action");
    try {
        if (action.equals("filter")) {
            list(request, response);
        }
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        sendErrorRedirect(request, response, e);
    }
}

private void list(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException,
    OKMAuth okmAuth = ContextWrapper.getContext().getBean(OKMAuth.class);

    String nameFilter = webUtils.getString(request, "nameFilter");
    String userFilter = webUtils.getString(request, "userFilter");

    // Getting lists
    List<Map<String, String>> leftDocsMaps = leftList(""); // Pending documents for all users
    List<Map<String, String>> rightDocsMaps = rightList(userFilter, nameFilter);
    List<String> users = new ArrayList<String>();
    // UserList
    for (CommonUser user : okmAuth.getUsers(null)) {
        users.add(user.getId());
    }

    ServletContext sc = getServletContext();
    sc.setAttribute("nameFilter", nameFilter);
    sc.setAttribute("leftDocsMaps", leftDocsMaps);
    sc.setAttribute("rightDocsMaps", rightDocsMaps);
    sc.setAttribute("users", users);
    sc.setAttribute("nameFilter", nameFilter);
    sc.setAttribute("userFilter", userFilter);
    sc.getRequestDispatcher("/sample/sample1/distributor.jsp").forward(request, response);
}

private List<Map<String, String>> leftList(String user) throws Exception {
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String uuid = okmRepository.getNodeUuid(null, "/okm:root");

    List<Map<String, String>> documentsMap = new ArrayList<Map<String, String>>();
    // To change
    List<Document> pendingDocuments = new ArrayList<Document>();
```

```
    for (Document doc : okmDocument.getChildren(null, uuid)) {
        Map<String, String> docMap = new HashMap<String, String>();
        docMap.put("author", doc.getAuthor());
        docMap.put("name", doc.getPath().substring(doc.getPath().lastIndexOf("/") + 1));
        docMap.put("uuid", doc.getUuid());
        docMap.put("path", doc.getPath());
        docMap.put("mimeType", doc.getMimeType());
        docMap.put("lastModified", df.format(doc.getLastModified().getTime()));
        documentsMap.add(docMap);
    }
    return documentsMap;
}

private List<Map<String, String>> rightList(String user, String nameFilter) throws Exception {
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String uuid = okmRepository.getNodeUuid(null, "/okm:root");

    List<Map<String, String>> documentsMap = new ArrayList<Map<String, String>>();
    for (Document doc : okmDocument.getChildren(null, uuid)) {
        boolean found = true;
        if (!user.equals("") && !user.equals(doc.getAuthor())) {
            found = false;
        }
        if (!nameFilter.equals("") && !doc.getPath().contains(nameFilter)) {
            found = false;
        }

        if (found) {
            Map<String, String> docMap = new HashMap<String, String>();
            docMap.put("author", doc.getAuthor());
            docMap.put("name", doc.getPath().substring(doc.getPath().lastIndexOf("/") + 1));
            docMap.put("uuid", doc.getUuid());
            docMap.put("path", doc.getPath());
            docMap.put("mimeType", doc.getMimeType());
            docMap.put("lastModified", df.format(doc.getLastModified().getTime()));
            documentsMap.add(docMap);
        }
    }
    return documentsMap;
}
%>
```

distributor.jsp

The main layer.

```
<%@ 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>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/sample/css/style.css" />
  <script type="text/javascript" src="<%=request.getContextPath() %>/js/jquery-1.7.1.min.js"></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/js/vanadium-min.js" ></script>
</head>
<body>
 <table width="100%">
  <tbody>
  <tr>
   <td valign="top" width="50%">
    <jsp:include page="/sample/sample1/full_list.jsp"></jsp:include>
```

```
    </td>
    <td valign="top" width="10"></td>
    <td valign="top" width="50%">
     <jsp:include page="/sample/sample1/filtered_list.jsp"></jsp:include>
    </td>
   </tr>
   </tbody>
  </table>
 </body>
 </html>
```

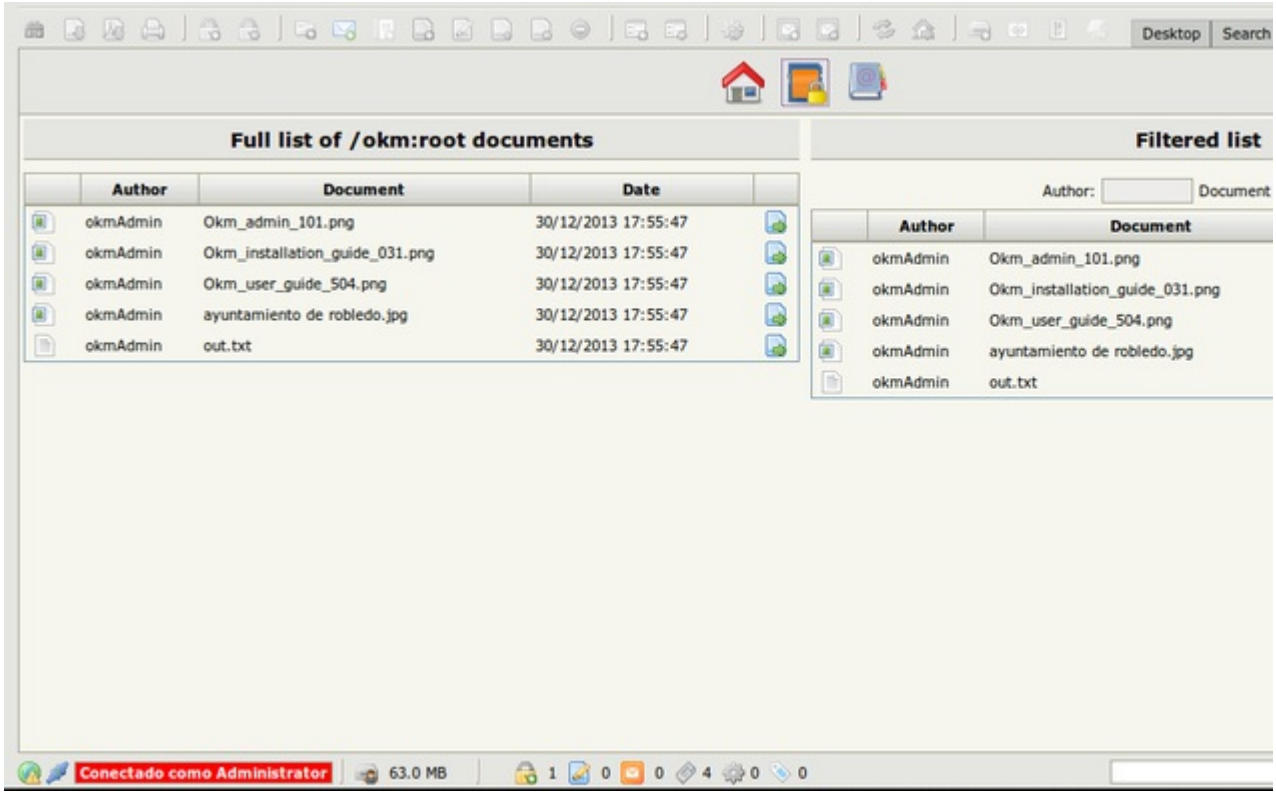
full_list.jsp

Shows the document list.

> *Pay attention to the JavaScript function "**go**", which calls parent.parent.jsOpenPathByUuid(uuid). The "**jsOpenPathByUuid**" is a GWT public function used to jump to a specific document, folder, etc. Here Exposed frontend JavaScript API you can find a complete list of JavaScript functions.*

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<script type="text/javascript">
 // Note the parent.parent call because iframe is into other iframe
    function go(uuid) {
        alert('Document UUID:'+uuid);
     parent.parent.jsOpenPathByUuid(uuid);
    }
</script>
<h1>Full list of /okm:root documents</h1>
<table class="results" style="white-space: nowrap;" cellpadding="3" width="100%">
    <thead>
      <tr>
        <th></th><th>Author</th><th>Document</th><th>Date</th><th></th>
      </tr>
    </thead>
    <tbody>
     <c:forEach var="document" items="${leftDocsMaps}" varStatus="row">
     <tr>
      <td valign="center"><img src="<%=request.getContextPath() %>/mime/${document.mimeType}"/></td>
      <td>${document.author}</td>
      <td>${document.name}</td>
      <td>${document.lastModified}</td>
      <td align="center">
       <a href="javascript:go('${document.uuid}')"><img src="<%=request.getContextPath() %>/sample/img/action/got
       </td>
     </tr>
     </c:forEach>
    </tbody>
</table>
```

Screenshot of sample1 view



Sample 2

Is composed of 4 files:

- *action.jsp (the main controller where the logic is set).*
- *add_form.jsp (form to upload a new document).*
- *distributor.jsp (the main layer with two columns at 50% with JavaScript).*
- *full_list.jsp (shows the full list with special JavaScript control for table height).*
- *pending.jsp (static content).*

action.jsp

The main controller.

```
<%@page import="java.util.Map"%>
<%@page import="java.util.HashMap"%>
<%@page import="java.util.List"%>
<%@page import="java.util.ArrayList"%>
<%@ page import="java.text.SimpleDateFormat"%>
<%@ page import="java.io.IOException"%>
<%@ page import="org.slf4j.Logger"%>
<%@ page import="org.slf4j.LoggerFactory"%>
<%@ page import="com.openkm.bean.Document"%>
<%@ page import="com.openkm.util.WebUtils"%>
<%@ page import="com.openkm.api.OKMDocument"%>
```

```
<%@ page import="com.openkm.api.OKMRepository"%>
<%@ page import="com.openkm.util.ContextWrapper"%>
<%@ page extends="com.openkm.extension.servlet.BaseServlet"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%!
private static Logger log = LoggerFactory.getLogger("com.openkm.sample");
private WebUtils webUtils = ContextWrapper.getContext().getBean(WebUtils.class);

public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletExcepti
    String action = webUtils.getString(request, "action");
    try {
        list(request, response);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        sendErrorRedirect(request, response, e);
    }
}

// Used by fileupload return and filter
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletExcept
    String action = webUtils.getString(request, "action");
    try {
        list(request, response);
    } catch (Exception e) {
        log.error(e.getMessage(), e);
        sendErrorRedirect(request, response, e);
    }
}

private void list(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, Exception {
    OKMDocument okmDocument = ContextWrapper.getContext().getBean(OKMDocument.class);
    OKMRepository okmRepository = ContextWrapper.getContext().getBean(OKMRepository.class);
    String uuid = okmRepository.getNodeUuid(null, "/okm:root");

    SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
    String docPath = (String) request.getAttribute("docPath");
    String nameFilter = webUtils.getString(request, "nameFilter");
    // Pending docs
    List<Map<String, String>> pendingDocMaps = new ArrayList<Map<String, String>>();
    for (Document doc : okmDocument.getChildren(null, uuid)) {
        Map<String, String> docMap = new HashMap<String, String>();
        docMap.put("name", doc.getPath().substring(doc.getPath().lastIndexOf("/") + 1));
        docMap.put("uuid", doc.getUuid());
        docMap.put("path", doc.getPath());
        docMap.put("mimeType", doc.getMimeType());
        docMap.put("lastModified", df.format(doc.getLastModified().getTime()));
        pendingDocMaps.add(docMap);
    }
    ServletContext sc = getServletContext();
    sc.setAttribute("docPath", docPath);
    sc.setAttribute("nameFilter", nameFilter);
    sc.setAttribute("pendingDocuments", pendingDocMaps);
    sc.getRequestDispatcher("/sample/sample2/distributor.jsp").forward(request, response);
}
 %>
```

distributor.jsp

The main layer.

> *Here, JavaScript is used to control the height and width of the layer.*

```
<%@ 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>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/sample/css/style.css" />
  <link rel="stylesheet" type="text/css" href="<%=request.getContextPath() %>/css/jquery.tablescroll.css"/>
  <script type="text/javascript" src="<%=request.getContextPath() %>/js/jquery-1.7.1.min.js"></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/js/vanadium-min.js" ></script>
  <script type="text/javascript" src="<%=request.getContextPath() %>/js/jquery.tablescroll.js"></script>
</head>
<body>
 <div id="leftPanel" style="float: left;">
  <jsp:include page="/sample/sample2/add_form.jsp"></jsp:include>
  <jsp:include page="/sample/sample2/full_list.jsp"></jsp:include>
 </div>
 <div id="centerPanel" style="float: left; width: 12px;">
  <div id="verticalSeparator" class="verticalSeparator"></div>
 </div>
 <div id="rightPanel" style="float: left;">
       <jsp:include page="/sample/sample2/pending.jsp"></jsp:include>
 </div>

 <script type="text/javascript">
  var width =  $(window).width()-12; // Deleting middle panel
  var height = $(window).height();

  // Setting height
  $('#verticalSeparator').height(height);
  $('#leftPanel').height(height);
  $('#rightPanel').height(height);
  // Setting width
  $('#leftPanel').width(parseInt(width/2));
  $('#centerPanel').width(12);
  $('#rightPanel').width((width-parseInt((width/2))));
 </script>
</body>
</html>
```

add_form.jsp

Form to upload a new document.

```
<%@ page import="com.openkm.frontend.client.constants.ui.UIFileUploadConstants" %>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<h2 id="headerTitle_new_register">Upload new file</h2>
<form enctype="multipart/form-data" method="post" action="<%=request.getContextPath() %>/frontend/FileUpload"
 <input type="hidden" name="path" value="/okm:root">
 <input type="hidden" name="action" value="<%=UIFileUploadConstants.ACTION_INSERT%>">
 <input type="hidden" name="message" value="">
 <input type="hidden" name="comment" value="">
 <input type="hidden" name="redirect" value="/sample/sample2/action.jsp">
 <table class="form">
    <tbody>
     <c:if test="${docPath != null && docPath != ''}">
     <tr>
      <td><div class="ok">Document sended</div></td>
     </tr>
  </c:if>
     <tr>
```

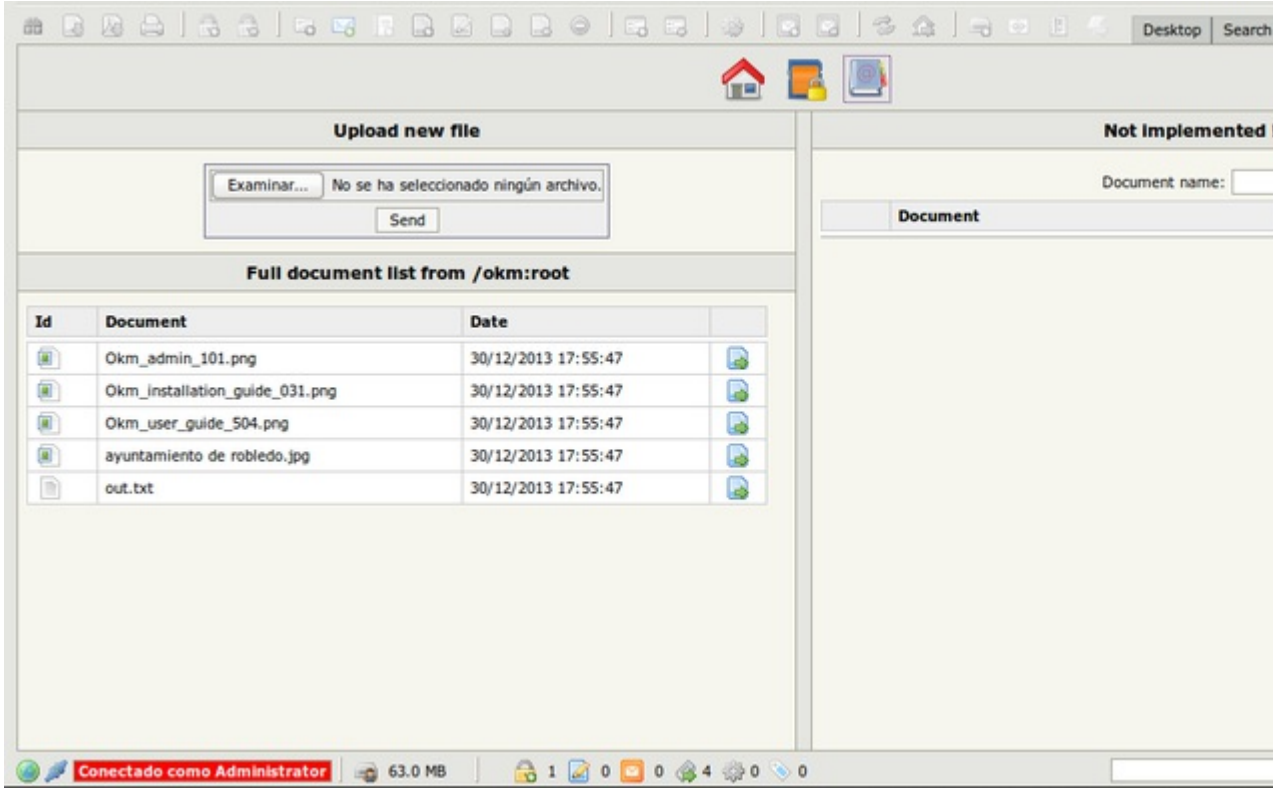
```
      <td>
       <div id="file_error" style="display:none; color : red;">File is mandatory.</div>
       <input type="file" name="file" class=":required;;file_error :only_on_blur">
      </td>
     </tr>
     <tr>
       <td align="center"><input type="submit" value="Send"></td>
     </tr>
   </tbody>
   </table>
</form>
```

full_list.jsp

Shows a list of documents.

> *Take a look at the end where JavaScript is used to control the table's scrolling.*

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<script type="text/javascript">
 // Note the parent.parent call because iframe is into other iframe
    function go(uuid) {
        alert('Document UUID:'+uuid);
     parent.parent.jsOpenPathByUuid(uuid);
    }
</script>
<h2 class="top">Full document list from /okm:root</h2>
<div align="center" style="padding: 0px 20px 0px 5px">
 <div class="tablescroll">
  <table id="docsPendingTable" width="100%" cellspacing="0" style="white-space: nowrap;">
      <thead>
        <tr>
          <td>Id</td><td>Document</td><td>Date</td><td></td>
        </tr>
      </thead>
      <tbody>
       <c:set var="first" value="true"></c:set>
       <c:forEach var="document" items="${pendingDocuments}" varStatus="row">
        <c:choose>
          <c:when test="${first==true}">
           <tr class="first">
          </c:when>
          <c:otherwise>
           <tr>
          </c:otherwise>
        </c:choose>
         <td valign="center"><img src="<%=request.getContextPath() %>/mime/${document.mimeType}"/></td>
         <td>${document.name}</td>
         <td>${document.lastModified}</td>
         <td align="center"><a href="javascript:go('${document.uuid}')"><img src="<%=request.getContextPath() %>/sa
        </tr>
        <c:set var="first" value="false"></c:set>
       </c:forEach>
      </tbody>
  </table>
 </div>
</div>

<script type="text/javascript">
```

```
//Calculating table height removing from total other elements height
var pendingTableHeight = $(window).height() - (parseInt($('#docsPendingTable').offset().top) + 35);
/*<![CDATA[*/
jQuery(document).ready(function($)
{
 $('#docsPendingTable').tableScroll({height:pendingTableHeight});
});
/*]]>*/
</script>
```

Screenshot of sample2 view