



Documentation for SDK for Java 4.11

## Table of Contents

Table of Contents	2
SDK for Java 4.11	12
License	12
Compatibility	12
Sample client	13
Jar sample	14
Basic concepts	16
Authentication	16
Accessing API	17
Class Hierarchy	20
Activity samples	23
Basics	23
Suggested code sample	23
Methods	23
findActivityLog	23
getActivityActions	24
Auth samples	26
Basics	26
Suggested code sample	26
Methods	26
login	27
login	27
logout	28
getRefreshToken	29
getGrantedUsersAndRoles	30
getGrantedRoles	30
getGrantedUsers	31
getRoles	32
getRolesByUser	32
getUsers	33
getUser	34
getUsersByRole	34
revokeRole	35
revokeUser	36
grantRole	37
grantUser	37
changeSecurity	38
overwriteSecurity	39
hasSecurityRecursive	40
hasTaskManagerAdmin	41
isAdmin	41
getSessionId	42
createUser	43
deleteUser	43
updateUser	44
createRole	44
deleteRole	45
updateRole	46
assignRole	46
removeRole	47
getProfiles	47
getUserProfile	48
setUserProfile	49
getUserToken	50
setUserPermissions	50
setRolePermissions	51

getUserTenants	52
setUserTenant	53
isLoginLowercase	53
isPasswordExpired	54
getToken	55
createPersonalAccessToken	55
<b>Bookmark samples</b>	<b>57</b>
Basics	57
Suggested code sample	57
Methods	57
getUserBookmarks	57
create	58
rename	58
delete	59
<b>Conversion samples</b>	<b>61</b>
Basics	61
Suggested code sample	61
Methods	61
doc2pdf	61
imageConvert	62
html2pdf	63
doc2txt	64
img2txt	65
barcode2txt	66
<b>Dashboard samples</b>	<b>67</b>
Basics	67
Suggested code sample	67
Methods	67
getUserCheckedOutDocuments	67
getUserLastModifiedDocuments	68
getUserLockedDocuments	69
getUserLockedRecords	70
getUserLockekFolders	72
getUserLockedMails	73
getUserSubscribedDocuments	74
getUserSubscribedFolders	75
getUserSubscribedRecords	76
getUserLastCreatedDocuments	77
getUserLastDocumentsNotesCreated	78
getUserLastCreatedFolders	79
getUserLastFoldersNotesCreated	80
getUserLastCreatedRecords	81
getUserLastRecordsNotesCreated	83
getUserLastDownloadedDocuments	84
getUserLastImportedMails	85
getUserLastMailsNotesCreated	86
getUserLastImportedMailAttachments	87
getUserSearches	88
findUserSearches	89
<b>Document samples</b>	<b>90</b>
Basics	90
Suggested code sample	90
Methods	90
create	90
create	91
create	92
delete	93
getProperties	93
getContent	94
getContentByVersion	95
getChildren	96
rename	97
setProperties	97

setLanguage	98
setTitle	99
checkout	99
cancelCheckout	100
forceCancelCheckout	101
isCheckedOut	102
checkin	102
checkin	103
purge	104
move	105
copy	105
getVersionHistorySize	106
isValid	107
getPath	108
getDetectedLanguages	108
extendedCopy	109
getExtractedText	110
setExtractedText	111
getThumbnail	111
createFromTemplate	112
updateFromTemplate	114
getAnnotations	115
getDifferences	116
getCheckedOut	116
setNodeClass	117
setDispositionStage	118
setDescription	118
createWizard	119
isOCRDataCaptureSupported	120
recognize	120
captureData	121
getNumberOfPages	122
getPageAsImage	122
getLiveEditRestrictedMimeTypes	123
liveEditCheckin	124
liveEditCancelCheckout	124
liveEditForceCancelCheckout	125
mergePdf	126
liveEditSetContent	127
isAttachment	127
isConvertibleToPDF	128
getPdf	129
saveAsPdf	129
webPageImport	130
setIndexable	131
getXRefs	132
refresh	132
<b>FilePlan samples</b>	<b>134</b>
<b>Basics</b>	<b>134</b>
Suggested code sample	134
<b>Methods</b>	<b>134</b>
getRootSections	134
getRootSectionsFilteredBySecurity	135
getChildrenSections	136
getChildrenSectionsFilteredBySecurity	136
getChildrenClasses	137
getChildrenClassesFilteredBySecurity	138
findNodeClassByPk	139
findElectronicRecordClasses	139
findElectronicRecordClassesFilteredBySecurity	140
findFilteredByCodeOrNameFilteredBySecurity	141
findSectionFiltered	142
getNodeClassBreadcrumb	143
findAllNodeClasses	143
getRootSectionsIdWithChildren	144
getChildrenSectionsIdByTenantWithChildren	145

<b>Folder samples</b>	<b>146</b>
Basics	146
Suggested code sample	146
Methods	146
create	146
getProperties	147
delete	147
rename	148
move	149
getChildren	149
isValid	150
getPath	151
copy	151
extendedCopy	152
getContentInfo	153
purge	154
setStyle	155
createMissingFolders	155
setDescription	156
createFromTemplate	156
<b>Import samples</b>	<b>159</b>
Basics	159
Suggested code sample	159
Methods	159
importDocument	159
importFolder	160
<b>Mail samples</b>	<b>162</b>
Basics	162
Suggested code sample	162
Methods	162
getProperties	162
delete	163
purge	163
rename	164
move	165
copy	165
extendedCopy	166
getChildren	167
isValid	168
getPath	169
createAttachment	169
deleteAttachment	170
getAttachments	170
sendMailWithAttachments	171
sendMailWithAttachments	172
importEml	173
importMsg	174
setTitle	175
sendMail	175
sendMail	176
sendMail	177
setDispositionStage	178
setDescription	178
getContent	179
createWizard	180
getThumbnail	181
getMailsPaginated	182
getAccounts	183
getMailMessages	184
addAccount	184
updateAccount	185
testAccount	186
deleteAccount	187
importMessages	187

createFilter	188
updateFilter	189
deleteFilter	190
createRule	190
updateRule	192
deleteRule	193
getFilterRules	194
forwardEmail	194
getPdf	195
saveAsPdf	196
<b>Node samples</b>	<b>198</b>
<b>Basics</b>	<b>198</b>
Suggested code sample	198
<b>Methods</b>	<b>198</b>
getNodeByUuid	198
getVersionHistory	199
restoreVersion	199
deleteVersion	200
purgeVersionHistory	201
maybePromotedAsRecord	201
promoteAsRecord	202
degradeRecord	202
isElectronicRecordPath	203
getElectronicRecordInPath	204
getChildrenNodesPaginated	204
getChildrenNodesPaginated	206
getChildrenNodesByCategoryPaginated	208
getChildrenNodesByCategoryPaginated	209
getBreadcrumb	210
subscribe	211
unsubscribe	212
importZip	212
unZip	213
exportZip	213
getNodesFiltered	214
evaluateDownloadZip	215
generateDownloadToken	216
restore	217
hasNodesLockedByOtherUser	217
setComment	218
getPaginatorConfig	218
getPaginatorPlugins	219
<b>Note samples</b>	<b>221</b>
<b>Basics</b>	<b>221</b>
Suggested code sample	221
<b>Methods</b>	<b>221</b>
add	221
get	222
delete	223
set	223
list	224
getNoteHistories	225
<b>Notification samples</b>	<b>226</b>
<b>Basics</b>	<b>226</b>
Suggested code sample	226
<b>Methods</b>	<b>226</b>
notify	226
<b>PDF samples</b>	<b>228</b>
<b>Basics</b>	<b>228</b>
Suggested code sample	228
<b>Methods</b>	<b>228</b>
getImage	228
split	229

extract	230
remove	231
rotate	232
insertPages	233
optimize	234
<b>Plugin samples</b>	<b>236</b>
<b>Basics</b>	<b>236</b>
Suggested code sample	236
<b>Methods</b>	<b>236</b>
executePluginPost	236
executePostPluginReturnFile	237
executePluginGet	239
executePluginAtGetReturnFile	240
<b>Property samples</b>	<b>243</b>
<b>Basics</b>	<b>243</b>
Suggested code sample	243
<b>Methods</b>	<b>243</b>
addCategory	243
removeCategory	244
addKeyword	244
removeKeyword	245
setEncryption	246
unsetEncryption	247
setSigned	247
isSigned	248
<b>PropertyGroup samples</b>	<b>250</b>
<b>Basics</b>	<b>250</b>
Suggested code sample	250
<b>Methods</b>	<b>250</b>
addGroup	251
removeGroup	252
getGroups	253
getAllGroups	254
getPropertyGroupForm	254
getPropertyGroupFormDefinition	255
getPropertyGroupFormDefinition	256
getPropertyGroupForm	257
setProperties	257
hasGroup	259
getRegisteredDefinition	260
registerDefinition	261
getSuggestions	261
getProperties	263
getGroupsByVersion	263
getPropertiesByVersion	264
getGroupByVersionForm	265
getGroup	266
getSuggestBoxKeyValue	266
getSuggestBoxKeyValuesFiltered	267
validateField	268
<b>Record samples</b>	<b>271</b>
<b>Basics</b>	<b>271</b>
Suggested code sample	271
<b>Methods</b>	<b>271</b>
create	271
getProperties	272
delete	272
purge	273
rename	274
move	274
copy	275
isValid	276
getChildren	276

setTitle	277
getPath	278
setNodeClass	278
setDispositionStage	279
setDescription	279
extendedCopy	280
createWizard	281
createFromTemplate	282
createMissingRecords	283
<b>Relation samples</b>	<b>285</b>
Basics	285
Suggested code sample	285
Methods	285
getRelationTypes	285
add	286
delete	287
getRelations	288
getRelationGroups	288
addGroup	289
addNodeToGroup	290
deleteGroup	290
getGroup	291
setGroupName	292
getAllRelationGroups	292
deleteItemFromGroup	293
<b>Report samples</b>	<b>295</b>
Basics	295
Suggested code sample	295
Methods	295
getReports	295
getReport	296
generateDownloadReportToken	297
execute	297
executeSql	298
save	299
saveSql	300
getSqlReports	301
<b>Repository samples</b>	<b>303</b>
Basics	303
Suggested code sample	303
Methods	303
getRootFolder	303
getTrashFolder	304
getTrashFolderBase	304
getTemplatesFolder	305
getPersonalFolder	306
getPersonalFolderBase	306
getMailFolder	307
getMailFolderBase	308
getCategoriesFolder	308
purgeTrash	309
getUpdateMessage	310
getRepositoryUuid	310
hasNode	311
getNodePath	311
getNodeUuid	312
getAppVersion	313
copyAttributes	313
executeScript	314
executeScript	315
executeSqlQuery	316
executeSqlQuery	317
executeHqlQuery	318
executeHqlQuery	319

getTranslations	320
getConfiguration	321
getChangeLog	322
getServerTime()	323
getAvailableLocales	323
getLicenseInfo()	324
getClusterUuid()	325
getIsFilePlan()	325
<b>Search samples</b>	<b>327</b>
<b>Basics</b>	<b>327</b>
Suggested code sample	330
<b>Methods</b>	<b>331</b>
find	331
find	332
findSimpleNodeBasePaginated	333
findSimpleNodeBasePaginated	334
findSimpleNodeBasePaginated	336
getCategorizedDocuments	337
saveSearch	338
updateSearch	339
getSearch	339
getAllSearches	340
deleteSearch	341
getSearchConfig	341
getSearchPlugins	342
findAllDefaultByNodeClass	343
findByQuery	343
findByQuery	344
findSimpleNodeBaseByQueryPaginated	345
findSimpleNodeBaseByQueryPaginated	347
findWithMetadata	348
findWithMetadata	349
findWithMetadataPaginated	351
findWithMetadataPaginated	352
getMimeType	354
csvExport	354
<b>Shard samples</b>	<b>356</b>
<b>Basics</b>	<b>356</b>
Suggested code sample	356
<b>Methods</b>	<b>356</b>
getShards	356
setCurrentShard	357
changeShard	357
<b>Stamp samples</b>	<b>359</b>
<b>Basics</b>	<b>359</b>
Suggested code sample	359
<b>Methods</b>	<b>359</b>
getAllStamps	359
getStampTextByPk	360
getStampImageByPk	360
getStampBarcodeByPk	361
calculateStampCoordinates	362
calculateStampExpressions	363
stampText	364
stampImage	365
stampBarcode	365
stampImageManually	366
stampTextCustom	367
stampImageCustom	368
calculateFlyImageDimensions	370
getPersonalStamps	370
getPersonalStampImage	371
<b>Task samples</b>	<b>373</b>

<b>Basics</b>	<b>373</b>
Suggested code sample	375
<b>Methods</b>	<b>376</b>
getAssigned	376
getActive	378
getFinished	379
getNotified	381
getStatus	383
getProjects	383
getTypes	384
getAssignedCount	385
getActiveCount	386
getFinishedCount	386
getNotifiedCount	387
create	388
update	390
getTask	392
delete	393
createProject	393
updateProject	394
deleteProject	395
getProject	395
getProjects	396
createType	397
updateType	397
deleteType	398
getType	399
getTypes	399
createStatus	400
updateStatus	401
deleteStatus	401
getStatus	402
getStatus	403
createNote	403
updateNote	404
deleteNote	404
getNotes	405
<b>UserConfig samples</b>	<b>407</b>
<b>Basics</b>	<b>407</b>
Suggested code sample	407
<b>Methods</b>	<b>407</b>
getConfig	407
setHome	408
<b>Wizard Samples</b>	<b>409</b>
<b>Basics</b>	<b>409</b>
Suggested code sample	409
<b>Methods</b>	<b>409</b>
findByUser	409
findAllByUser	410
findByUserAndUuid	411
hasWizardByUserAndNode	411
deleteAll	412
postponeNode	412
cancelPostponeNode	413
deleteByNode	414
addDigitalSignature	414
removeDigitalSignature	415
addShowWizardCategories	415
removeShowWizardCategories	416
addShowWizardKeywords	417
removeShowWizardKeywords	417
addShowWizardOCRDataCapture	418
removeShowWizardOCRDataCapture	418
addGroup	419

removeGroup	420
addWorkflow	420
removeWorkflow	421
setAutostart	422
<b>Workflow samples</b>	<b>424</b>
Basics	424
Suggested code sample	424
Methods	424
registerProcessDefinition	424
deleteProcessDefinition	425
getProcessDefinition	425
runProcessDefinition	426
findProcessInstances	427
findAllProcessDefinitions	428
findLatestProcessDefinitions	429
findLastProcessDefinition	429
getProcessInstance	430
findUserTaskInstances	431
findTaskInstances	431
setTaskInstanceValues	432
getTaskInstance	433
startTaskInstance	434
setTaskInstanceActorId	434
endTaskInstance	435
getProcessDefinitionForms	436
getProcessDefinitionImage	437
findPooledTaskInstances	437
findProcessInstancesByNode	438
getSuggestBoxKeyValue	439
getSuggestBoxKeyValuesFiltered	439
<b>Purchase workflow sample</b>	<b>441</b>
Process image	441
Process definition	441
Process handlers	442
Form definition	442

## SDK for Java 4.11

OpenKM SDK for Java is a set of software development tools that allows the creation of applications for OpenKM. The OpenKM SDK for Java includes a Webservices library.

This Webservices library is a complete API layer to access OpenKM through REST Webservices and provides complete compatibility between the OpenKM REST Webservices versions minimizing the changes in your code.

## License



SDK for Java is licensed under the terms of the [EULA - OpenKM SDK End User License Agreement](#) and published by OpenKM Knowledge Management System S.L.

This program is distributed WITHOUT ANY WARRANTY not even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the [EULA - OpenKM SDK End User License Agreement](#) for more details.

## Compatibility



SDK for Java version 4.11 should be used:

- From OpenKM Professional version 8.1.14

## Sample client

You can make use of the OpenKM Maven Repository and the right SDK version. This is a sample pom.xml configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.openkm.sample</groupId>
  <artifactId>sample</artifactId>
  <version>1.0-SNAPSHOT</version>

  <repositories>
    <repository>
      <id>openkm.com</id>
      <name>OpenKM Maven Repository</name>
      <url>https://maven.openkm.com</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>com.openkm</groupId>
      <artifactId>sdk4j</artifactId>
      <version>4.11</version>
    </dependency>
  </dependencies>
</project>
```

Your first class:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Folder fld : ws.folder.getChildren("4f873d10-654e-4d99-a94f-15466e30a0f6")) {
                System.out.println("Folder -> " + fld.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Jar sample



If you use "**maven-assembly-plugin**" rather "**maven-shade-plugin**" you will get an error "**missing MessageBodyWriter**" while uploading a new file across the SDK.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.openkm.sample</groupId>
  <artifactId>sample</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <java.compiler>11</java.compiler>
    <sdk4j.version>4.11</sdk4j.version>
    <maven-compiler-plugin.version>3.1</maven-compiler-plugin.version>
    <maven-shade-plugin.version>2.4.3</maven-shade-plugin.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <repositories>
    <repository>
      <id>openkm.com</id>
      <name>OpenKM Maven Repository</name>
      <url>https://maven.openkm.com</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>com.openkm</groupId>
      <artifactId>sdk4j</artifactId>
      <version>${sdk4j.version}</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven-compiler-plugin.version}</version>
        <configuration>
          <source>${java.compiler}</source>
          <target>${java.compiler}</target>
          <encoding>${project.build.sourceEncoding}</encoding>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>${maven-shade-plugin.version}</version>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
            <configuration>
              <transformers>
                <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer"
```

```
<mainClass>com.openkm.Main</mainClass>
</transformer>
<transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
  <resource>META-INF/services/javax.ws.rs.ext.MessageBodyWriter</resource>
</transformer>
</transformers>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>
</project>
```

## Basic concepts

### Authentication

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Repository methods from "**repository**" class as is shown below:

```
ws.repository.getAppVersion();
```

```
package com.openkm;  
  
import com.openkm.sdk4j.OKMWebservicesFactory;  
import com.openkm.sdk4j.exception.DatabaseException;  
import com.openkm.sdk4j.exception.RepositoryException;  
import com.openkm.sdk4j.exception.UnknowException;  
import com.openkm.sdk4j.exception.WebserviceException;  
import com.openkm.sdk4j.impl.OKMWebservices;  
  
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);  
            System.out.println(ws.repository.getAppVersion());  
        } catch (RepositoryException e) {  
            e.printStackTrace();  
        } catch (DatabaseException e) {  
            e.printStackTrace();  
        } catch (UnknowException e) {  
            e.printStackTrace();  
        } catch (WebserviceException e) {  
            e.printStackTrace();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

}

## Accessing API

OpenKM API classes are under com.openkm package, as can shown at this [Javadoc API summary](#).



At main url <http://docs.openkm.com/javadoc/> you'll see all available Javadoc documentation.

At the moment of writing this page the actual OpenKM version was 7.1.5 what can change with time.



There is a direct correspondence between the classes and methods implemented at com.openkm.api packages and available from SDK for Java.

OpenKM API classes:

OpenKM API class	Description	Supported	Implementation	Interface
<b>OKMActivity</b>	Manages the activity log.	Yes	ActivityImpl.java	BaseActivity.java
<b>OKMAuth</b>	Manages security and users. For example add or remove grants on a node, create or modify users or getting the profiles.	Yes	AuthImpl.java	BaseAuth.java
<b>OKMBookmark</b>	Manages the user bookmarks.	Yes	BookmarkImpl.java	BaseBookmark.java
<b>OKMDashboard</b>	Manages all data shown at dashboard.	Yes	DashboardImpl.java	BaseDashboard.java
<b>OKMDocument</b>	Manages documents. For example create, move or delete a document.	Yes	DocumentImpl.java	BaseDocument.java

<b>OKMFolder</b>	Manages folders. For example create, move or delete a folder.	Yes	FolderImpl.java	BaseFolder.java
<b>OKMMail</b>	Manages mails. For example create, move or delete a mails.	Yes	MailImpl.java	BaseMail.java
<b>OKMNode</b>	Manages general purpose node actions.	Yes	NodeImpl.java	NodeBase.java
<b>OKMNote</b>	Manages notes on any node type. For example create, edit or delete a note on a document, folder, mail or record.	Yes	NoteImpl.java	BaseNote.java
<b>OKMNotification</b>	Manages notifications. For example add or remove subscriptions on a document or a folder.	Yes	NotificationImpl.java	BaseNotification.java
<b>OKMProperty</b>	Manages categories and keywords. For example add or remove keywords on a document, folder, mail or record.	Yes	PropertyImpl.java	BaseProperty.java
<b>OKMPropertyGroup</b>	Manages metadata. For example add metadata group, set metadata fields.	Yes	PropertyGroupImpl.java	BasePropertyGroup.java

<b>OKMRecord</b>	Manages records. For example create, move or delete a record.	Yes	RecordImpl.java	BaseRecord.java
<b>OKMRelation</b>	Manages relations between nodes. For example create a relation ( parent-child ) between two documents.	Yes	RelationImpl.java	BaseRelation.java
<b>OKMRepository</b>	A lot of options related with the repository. For example get the properties of main root node ( /okm:root ).	Yes	RepositoryImpl.java	BaseRepository.java
<b>OKMReport</b>	Manages reports. For example execute reports.	Yes	ReportImpl.java	BaseReport.java
<b>OKMSearch</b>	Manages search feature. For example manage saved queries or perform a new query to the repository.	Yes	SearchImpl.java	BaseSearch.java
<b>OKMStamp</b>	Manages stamp.	Yes	StampImpl.java	BaseStamp.java
<b>OKMStats</b>	General stats of the repository.	No		
<b>OKMTask</b>	Manages tasks. For	Yes	TaskImpl.java	BaseTask.java


	example create a new task.			
<b>OKMUserConfig</b>	Manages user home configuration.	No		
<b>OKMWorkflow</b>	Manages workflows. For example execute a new workflow.	Yes	WorkflowImpl.java	BaseWorkflow.java

## Class Hierarchy

Packages detail:

Name	Description
<b>com.openkm</b>	<p>The <b>com.openkm.OKMWebservicesFactory</b> that returns an <b>com.openkm.OKMWebservices</b> object which implements all interfaces.</p> <pre>OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);</pre>
<b>com.openkm.sdk4j.bean</b>	Contains all classes result of unmarshalling REST objects.
<b>com.openkm.sdk4j.definition</b>	<p>All interface classes:</p> <ul style="list-style-type: none"> <li>com.openkm.sdk4j.definition.BaseActivity</li> <li>com.openkm.sdk4j.definition.BaseAuth</li> <li>com.openkm.sdk4j.definition.BaseBookmark</li> <li>com.openkm.sdk4j.definition.BaseConversion</li> <li>com.openkm.sdk4j.definition.BaseDashboard</li> <li>com.openkm.sdk4j.definition.BaseFilePlan</li> <li>com.openkm.sdk4j.definition.BaseDocument</li> <li>com.openkm.sdk4j.definition.BaseFolder</li> <li>com.openkm.sdk4j.definition.BaseMail</li> <li>com.openkm.sdk4j.definition.BaseNode</li> <li>com.openkm.sdk4j.definition.BaseNote</li> </ul>

	<ul style="list-style-type: none"><li>• com.openkm.sdk4j.definition.BaseNotification</li><li>• com.openkm.sdk4j.definition.BasePlugin</li><li>• com.openkm.sdk4j.definition.BaseProperty</li><li>• com.openkm.sdk4j.definition.BasePropertyGroup</li><li>• com.openkm.sdk4j.definition.BaseRecord</li><li>• com.openkm.sdk4j.definition.BaseRelation</li><li>• com.openkm.sdk4j.definition.BaseRepository</li><li>• com.openkm.sdk4j.definition.BaseSearch</li><li>• com.openkm.sdk4j.definition.BaseStamp</li><li>• com.openkm.sdk4j.definition.BaseTask</li><li>• com.openkm.sdk4j.definition.BaseWorkflow</li></ul>
<b>com.openkm.sdk4j.impl</b>	<p>All interface implementation classes:</p> <ul style="list-style-type: none"><li>• com.openkm.sdk4j.impl.ActivityImpl</li><li>• com.openkm.sdk4j.impl.AuthImpl</li><li>• com.openkm.sdk4j.impl.BookmarkImpl</li><li>• com.openkm.sdk4j.impl.ConversionImpl</li><li>• com.openkm.sdk4j.impl.DashboardImpl</li><li>• com.openkm.sdk4j.impl.DocumentImpl</li><li>• com.openkm.sdk4j.impl.FileplanImpl</li><li>• com.openkm.sdk4j.impl.FolderImpl</li><li>• com.openkm.sdk4j.impl.MailImpl</li><li>• com.openkm.sdk4j.impl.NodeImpl</li><li>• com.openkm.sdk4j.impl.NoteImpl</li><li>• com.openkm.sdk4j.impl.NotificationImpl</li><li>• com.openkm.sdk4j.impl.PropertyGroupImpl</li><li>• com.openkm.sdk4j.impl.PropertyImpl</li><li>• com.openkm.sdk4j.impl.RecordImpl</li><li>• com.openkm.sdk4j.impl.RelationImpl</li><li>• com.openkm.sdk4j.impl.RepositoryImpl</li><li>• com.openkm.sdk4j.impl.SearchImpl</li><li>• com.openkm.sdk4j.impl.StampImpl</li><li>• com.openkm.sdk4j.impl.taskImpl</li><li>• com.openkm.sdk4j.impl.WorkflowImpl</li></ul>

<b>com.openkm.sdk4j.util</b>	A couple of utilities. <div data-bbox="539 365 1401 470" style="border: 1px dashed #add8e6; padding: 5px;"> The class <code>com.openkm.sdk4j.util.ISO8601</code> should be used to set and parse metadata date fields.</div>
<b>com.openkm.sdk4j.exception</b>	All exception classes.

## Activity samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Activity methods from "**activity**" class as is shown below:

```
ActivityList results = ws.activity.findActivityLog(0, 20, beginDate, endDate, user, "", item);
```

### Methods

#### findActivityLog

Description:

Method	Return values	Description
<b>findActivityLog(int page, int length, Calendar beginDate, Calendar endDate, String user, String action, String item)</b>	<b>ActivityList</b>	Returns a list of all activity log by date, user, action and item.
The value of the <b>item</b> is the UUID of the node ( document, folder, mail or record ).		

Example:

```
package com.openkm;
import java.util.Calendar;
```

```

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Activity;
import com.openkm.sdk4j.bean.ActivityList;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            Calendar beginDate = Calendar.getInstance();
            beginDate.add(Calendar.MONTH, -1);
            Calendar endDate = Calendar.getInstance();
            String item = "f84a2e1f-a858-4e53-9c09-36519d903782";

            ActivityList results = ws.activity.findActivityLog(0, 20, beginDate, endDate, user, "", item);
            for(Activity activity: results.getActivities()) {
                System.out.println(activity);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getActivityActions

Description:

Method	Return values	Description
<b>getActivityActions()</b>	<b>List&lt;String&gt;</b>	Returns a list of all the activity log actions.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (String action : ws.activity.getActivityActions()) {
                System.out.println(action);
            }
        }
    }
}

```

```
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

## Auth samples

### Basics

The class `com.openkm.sdk4j.bean.Permission` contains permission values ( `READ`, `WRITE`, etc. ). You should use it in combination with methods that are changing or getting security grants.



To set `READ` and `WRITE` access you should do:

```
int permission = Permission.READ + Permission.WRITE;
```

To check if you have permission access you should do:

```
// permission is a valid integer value
if ((permission | Permission.WRITE) = Permission.WRITE) {
    // Has WRITE grants.
}
```



Example of uuid:

- Using UUID -> "`c41f9ea0-0d6c-45da-bae4-d72b66f42d0f`";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "`login`". You can access the "`login`" method from webservice object "`ws`" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Auth methods from "`auth`" class as is shown below:

```
Map<String, Integer> grants = ws.auth.getGrantedRoles("373bcdd0-c082-4e7b-addd-e10ef813946e");
```

### Methods

**login**

Description:

Method	Return values	Description
<b>login(String user, String password)</b>	String	Login process return authentication token.

**i** Login token by **default** have an **expiration time of 24h**.  
 After executing the login method, all the next calls will use automatically the authentication token.  
 Each time login method is executed the login token is replaced by newer.

**⚠** When user login into the application the first time, is called internally a method what creates user specific folders, like /okm:trash/userId etc.  
 From API point of view, this method should be only executed first time user accessing from API, for creating specific user folder structure.  
 Otherwise you can get errors, for example PathNotExistsException /okm:trash/userId when deleting objects, because the folders has not been created.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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 {
            System.out.println("Token: " + ws.login(user, password));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**login**

Description:

Method	Return values	Description
--------	---------------	-------------

<b>login(String user, String password, int expiration, boolean restrictIP)</b>	String	Login process return authentication token.
--	--------	--

**i** Login token by default have an **expiration** time of variable **expiration value in days**.

When **restrictIP** is true, the token **only will work from the source IP address**.

After executing the login method, all the next calls will use automatically the authentication token.

Each time login method is executed the login token is replaced by newer.

**⚠** When user login into the application the first time, is called internally a method what creates user specific folders, like /okm:trash/userId etc.

From API point of view, this method should be only executed first time user accessing from API, for creating specific user folder structure.

Otherwise you can get errors, for example PathNotExistsException /okm:trash/userId when deleting objects, because the folders has not been created.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        int days = 7;
        boolean restrictIp = true;


        try {
            System.out.println("Token: " + ws.login(user, password, days, restrictIp));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**logout**

Description:

Method	Return values	Description

<b>logout()</b>	void	Execute logout method in OpenKM side.
-----------------	------	---------------------------------------


Thee authentication token will be reset.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.logout();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getRefreshToken**

Description:

Method	Return values	Description
<b>getRefreshToken()</b>	void	Refresh the current authentication token.


When login, it is set in the OKMWebservice an authentication token. By default, the authentication token expires in 24 hours. This method allows the replacement of the current with a new fresh one.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    // getRefreshToken
    String token = ws.getRefreshToken();
    System.out.println("New Token: " + token);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### getGrantedUsersAndRoles

Description:

Method	Return values	Description
<b>getGrantedUsersAndRoles(String uuid)</b>	<b>GrantedUsersAndRolesItem</b>	Return the granted users and roles of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.GrantedUsersAndRolesItem;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            GrantedUsersAndRolesItem grants = ws.auth.getGrantedUsersAndRoles("3c68b3a1-c65c-4b1e-84b5-9ce27
            for (String user : grants.getGrantedUsers().keySet()) {
                System.out.println(user + "->" + grants.getGrantedUsers().get(user));
            }
            for (String role : grants.getGrantedRoles().keySet()) {
                System.out.println(role + "->" + grants.getGrantedRoles().get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getGrantedRoles

Description:

Method	Return values	Description
<b>getGrantedRoles(String uuid)</b>	<b>Map&lt;String, Integer&gt;</b>	Return the granted roles of a node.

Example:

```

package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, Integer> grants = ws.auth.getGrantedRoles("373bcdd0-c082-4e7b-addd-e10ef813946e");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getGrantedUsers

Description:

Method	Return values	Description
<b>getGrantedUsers(String uuid)</b>	<b>Map&lt;String, Integer&gt;</b>	Return the granted users of a node.

Example:

```

package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {

```

```
String host = "http://localhost:8080/openkm";
String username = "okmAdmin";
String password = "admin";
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

try {
    ws.login(username, password);
    Map<String, Integer> grants = ws.auth.getGrantedUsers("373bcdd0-c082-4e7b-addd-e10ef813946e");
    for (String user : grants.keySet()) {
        System.out.println(user + "->" + grants.get(user));
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**getRoles**

Description:

Method	Return values	Description
<b>getRoles(boolean showAll)</b>	<b>List&lt;String&gt;</b>	Return the list of all the roles.
When showAll variable is set to true return all the roles - enabled and disabled - otherwise only returns the enabled.		

Example:

```
package com.openkm;

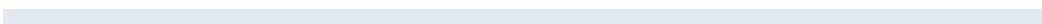
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
        try {
            ws.login(username, password);
            for (String role : ws.auth.getRolesByUser("okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getRolesByUser**

Description:



Method	Return values	Description
<b>getRolesByUser(String user)</b>	<b>List&lt;String&gt;</b>	Return the list of all the roles assigned to a user.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
        try {
            ws.login(username, password);
            for (String role : ws.auth.getRolesByUser("okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUsers**

Description:

Method	Return values	Description
<b>getUsers(boolean showAll)</b>	<b>List&lt;CommonUser&gt;</b>	Return the list of all the users.
When showAll variable is set to true return all the users - enabled and disabled - otherwise only returns the enabled.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.CommonUser;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        for (CommonUser commonUser : ws.auth.getUsers(true)) {
            System.out.println(commonUser);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getUser**

Description:

Method	Return values	Description
<b>getUser(String userId)</b>	<b>CommonUser</b>	Return all user data

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.CommonUser;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            CommonUser commonUser = ws.auth.getUser("okmAdmin");
            System.out.print(commonUser);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getUsersByRole**

Description:

Method	Return values	Description
<b>getUsersByRole(String role)</b>	<b>List&lt;CommonUser&gt;</b>	Return the list of all the users who have assigned a role.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.CommonUser;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (CommonUser commonUser : ws.auth.getUsersByRole("ROLE_ADMIN")) {
                System.out.println(commonUser);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**revokeRole**

Description:

Method	Return values	Description
<b>revokeRole(String uuid, String roleId, int permissions, boolean recursive)</b>	<b>void</b>	Remove role grant on a node.
<p>The parameter recursive only has sense when the uuid is a folder or a record node.</p> <p>When parameter recursive is true, the change will be applied to the node and its descendants.</p>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    // Remove ROLE_USER write grants at the node but not descendants
    ws.auth.revokeRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", Permission.ALL_GRANTS);

    // Remove all ROLE_ADMIN grants to the node and descendants
    ws.auth.revokeRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_ADMIN", Permission.ALL_GRANTS);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**revokeUser**

Description:

Method	Return values	Description
<b>revokeUser(String uuid, String user, int permissions, boolean recursive)</b>	<b>void</b>	Remove user grant on a node.
<p>The parameter recursive only has sense when the uuid is a folder or a record node.</p> <p>When parameter recursive is true, the change will be applied to the node and descendants.</p>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Remove sochoa write grants at the node but not descendants
            ws.auth.revokeUser("373bcdd0-c082-4e7b-addd-e10ef813946e", "sochoa", Permission.ALL_GRANTS, false);

            // Remove all okmAdmin grants at the node and descendants
            ws.auth.revokeUser("373bcdd0-c082-4e7b-addd-e10ef813946e", "okmAdmin", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}

```

**grantRole**

Description:

Method	Return values	Description
<b>grantRole(String uuid, String role, int permissions, boolean recursive)</b>	<b>void</b>	Add role grant on a node.
<p>The parameter recursive only has sense when the uuid is a folder or a record node.</p> <p>When parameter recursive is true, the change will be applied to the node and descendants.</p>		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Add ROLE_USER write grants at the node but not descendants
            ws.auth.grantRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", Permission.ALL_GRANTS, f

            // Add all ROLE_ADMIN grants to the node and descendants
            ws.auth.grantRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_ADMIN", Permission.ALL_GRANTS, f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**grantUser**

Description:

Method	Return values	Description
<b>grantUser(String uuid, String user, int permissions, boolean recursive)</b>	<b>void</b>	Add user grant on a node.

The parameter recursive only has sense when the uuid is a folder or record node.

When parameter recursive is true, the change will be applied to the node and descendants.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Add sochoa write grants at the node but not descendants
            ws.auth.grantUser("4f873d10-654e-4d99-a94f-15466e30a0f6", "sochoa", Permission.ALL_GRANTS, false);

            // Add all okmAdmin grants at the node and descendants
            ws.auth.grantUser("4f873d10-654e-4d99-a94f-15466e30a0f6", "okmAdmin", Permission.ALL_GRANTS, true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**changeSecurity**

Description:

Method	Return values	Description
<b>changeSecurity(String uuid, ChangeSecurity changeSecurity)</b>	<b>void</b>	Change the security of a node.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ChangeSecurity;
import com.openkm.sdk4j.bean.GrantedRole;
import com.openkm.sdk4j.bean.GrantedUser;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;
    
```

```

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);

            List<GrantedUser> guList = new ArrayList<>();
            GrantedUser gu = new GrantedUser();
            gu.setUser("sochoa");
            gu.setPermissions(Permission.ALL_GRANTS);
            guList.add(gu);

            List<GrantedRole> grList = new ArrayList<>();
            GrantedRole gr = new GrantedRole();
            gr.setRole("ROLE_TEST");
            gr.setPermissions(Permission.READ | Permission.WRITE);
            grList.add(gr);


            ChangeSecurity cs = new ChangeSecurity();
            cs.setGrantedUsersList(guList);
            cs.setGrantedRolesList(grList);
            cs.setRecursive(true);
            ws.auth.changeSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", cs);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**overwriteSecurity**

Description:

Method	Return values	Description
<b>overwriteSecurity(String uuid, ChangeSecurity changeSecurity)</b>	<b>void</b>	Overwrite the security of a node.



The ChangeSecurity object is used in changeSecurity and overwriteSecurity methods. Although set values in the variables revokeUsers and revokeRoles of the ChangeSecurity object, these values will not be taken in consideration while overwritten the security.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;

```

```

import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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);
            List<GrantedUser> guList = new ArrayList<>();
            GrantedUser gu = new GrantedUser();
            gu.setUser("sochoa");
            gu.setPermissions(Permission.ALL_GRANTS);
            guList.add(gu);

            List<GrantedRole> grList = new ArrayList<>();
            GrantedRole gr = new GrantedRole();
            gr.setRole("ROLE_TEST");
            gr.setPermissions(Permission.READ | Permission.WRITE);
            grList.add(gr);

            ChangeSecurity cs = new ChangeSecurity();
            cs.setGrantedUsersList(guList);
            cs.setGrantedRolesList(grList);
            cs.setRecursive(true);
            ws.auth.overrideSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", cs);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**hasSecurityRecursive**

Description:

Method	Return values	Description
hasSecurityRecursive()	Boolean	Check if the user has grants to propagate the security recursively.


The configuration parameter named "**default.security.recursive.role**" is used to indicate the role.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

```


```
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);
            System.out.println("hasSecurityRecursive = " + ws.auth.hasSecurityRecursive());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**hasTaskManagerAdmin**

Description:

Method	Return values	Description
hasTaskManagerAdmin()	Boolean	Check if the user is a member of the role task manager admin

 The configuration parameter named "**default.task.manager.admin.role**" is used to indicate the role.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("hasTaskManagerAdmin = " + ws.auth.hasTaskManagerAdmin());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**isAdmin**

Description:

Method	Return values	Description
isAdmin()	Boolean	Check if the authenticated user is a member of the administrators.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("Is Administrator " + ws.auth.isAdmin());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getSessionId

Description:

Method	Return values	Description
<b>getSessionId()</b>	<b>String</b>	Get http session id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.auth.getSessionId());
        }
    }
}

```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**createUser**

Description:

Method	Return values	Description
<b>createUser(String userId, String password, String email, String name, boolean active)</b>	<b>void</b>	Create a new user.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.createUser("test", "password.2016", "test@mail.com", "User test", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**deleteUser**

Description:

Method	Return values	Description
<b>deleteUser(String userId)</b>	<b>void</b>	Delete a user.

Example:

```

package com.openkm;

```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.deleteUser("test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**updateUser**

Description:

Method	Return values	Description
<b>updateUser(String userId, String password, String email, String name, boolean active)</b>	<b>void</b>	Update a user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.updateUser("test", "newpassword", "test@mail.com", "Test", false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**createRole**

Description:

Method	Return values	Description
<b>createRole(String roleId, boolean active)</b>	<b>void</b>	Create a new role.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.createRole("ROLE_TEST", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### deleteRole

Description:

Method	Return values	Description
<b>deleteRole(String roleId)</b>	<b>void</b>	Delete a role.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

```

```

        ws.auth.deleteRole("ROLE_TEST");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**updateRole**

Description:

Method	Return values	Description
<b>updateRole(String roleId, boolean active)</b>	<b>void</b>	Update a role.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.updateRole("ROLE_TEST", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**assignRole**

Description:

Method	Return values	Description
<b>assignRole(String userId, String roleId)</b>	<b>void</b>	Assign role to a user.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

```

```
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);
            ws.auth.assignRole("test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**removeRole**

Description:

Method	Return values	Description
<b>removeRole(String userId, String roleId)</b>	<b>void</b>	Remove a role from a user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.auth.removeRole("test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getProfiles**

Description:

Method	Return values	Description

<b>getProfiles(boolean filterByActive)</b>	<b>List&lt;Profile&gt;</b>	Return the list of all profiles.
--	----------------------------	----------------------------------

The parameter filterByActive when enabled the method will return only the active profiles, otherwise will return all available profiles.

**i** Each user has assigned one profile that enables more or less of the OpenKM UI features.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Profile;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Profile profile : ws.auth.getProfiles(true)) {
                System.out.println(profile.getName());
                System.out.println(profile.getActive());
                System.out.println(profile.getPrfMisc());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUserProfile**

Description:

Method	Return values	Description
<b>getUserProfile(String userId)</b>	<b>Profile</b>	Return the profile assigned to a user.

**i** Each user has assigned one profile that enables more or less of the OpenKM UI features.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Profile;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Profile profile = ws.auth.getUserProfile("okmAdmin");
            System.out.println(profile.getName());
            System.out.println(profile.getActive());
            System.out.println(profile.getPrfMisc());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**setUserProfile**

Description:

Method	Return values	Description
<b>setUserProfile(String userId, long profileId)</b>	<b>void</b>	Change the assigned profile to a user.


Each user has assigned one profile that enables more or less of the OpenKM UI features.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Profile;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Set the profile named "default" to the user
        }
    }
}
    
```

```

        for (Profile profile : ws.auth.getProfiles(true)) {
            if (profile.getName().equals("default")) {
                ws.auth.setUserProfile("test", profile.getId());
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

**getUserToken**

Description:

Method	Return values	Description
<b>getUserToken(String userId)</b>	<b>String</b>	Return the token to a user.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String token = ws.auth.getUserToken("okmAdmin");
            System.out.println(token);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setUserPermissions**

Description:

Method	Return values	Description
<b>setUserPermissions(String uuid, String userId, int permissions, boolean recursive)</b>	<b>void</b>	Update user permissions on a node.

The parameter recursive only has sense when the uuid is a folder or record node.

When parameter recursive is true, the change will be applied to the node and descendants.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Update permissions of sochoa at the node but not descendants
            ws.auth.setUserPermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "sochoa", Permission.READ + Pe

            // Update permissions of okmAdmin at the node and descendants
            ws.auth.setUserPermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "okmAdmin", Permission.ALL_GF
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**setRolePermissions**

Description:

Method	Return values	Description
<b>setRolePermissions(String uuid, String roleId, int permissions, boolean recursive)</b>	<b>void</b>	Update role permissions on a node.

The parameter recursive only has sense when the uuid is a folder or record node.

When parameter recursive is true, the change will be applied to the node and descendants.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
    
```

```

import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Update permissions of ROLE_USER at the node but not descendants
            ws.auth.setRolePermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_USER", Permission.REAL

            // Update permissions of ROLE_ADMIN at the node and descendants
            ws.auth.setRolePermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_ADMIN", Permission.ALL
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getUserTenants

Description:

Method	Return values	Description
<b>getUserTenants()</b>	<b>List&lt;Tenant&gt;</b>	Return the list of all tenants.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Tenant;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<Tenant> tenants = ws.auth.getUserTenants();
            for (Tenant tenant : tenants) {
                System.out.println(tenant);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

```
}
}
```

**setUserTenant**

Description:

Method	Return values	Description
<b>setUserTenant(long tenantId)</b>	<b>void</b>	Change the assigned tenant to a user.


User might have access to several tenants but only have one assigned.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long tenantId = 1;
            ws.auth.setUserTenant(tenantId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**isLoginLowercase**

Description:

Method	Return values	Description
<b>isLoginLowercase()</b>	<b>void</b>	Return true when user is must be set in lowercase in login.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.auth.isLoginLowercase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**isPasswordExpired**

Description:

Method	Return values	Description
<b>isPasswordExpired()</b>	<b>void</b>	True when the password of the user has expired.


By default the password expiration is disabled. Take a look at the configuration parameter named **"user.password.expiration"** in the [Security configuration parameters](#) section to enable.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("Password expired: " + ws.auth.isPasswordExpired());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### getToken

Description:

Method	Return values	Description
<b>getToken()</b>	void	Get a new authentication token

**i** Authentication token expires after 24 hours or later ( it depends on how was created ). This method helps in refreshing the current token.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.getToken();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### createPersonalAccessToken

Description:

Method	Return values	Description
<b>createPersonalAccessToken(String name, Calendar expiration)</b>	void	Create a personal access authentication token

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
    
```

```
import com.openkm.sdk4j.impl.OKMWebservices;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Calendar;

public class Test {

    private static final Logger log = LoggerFactory.getLogger(Test.class);

    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);
            // createPersonalAccessToken
            Calendar expiration = Calendar.getInstance();
            expiration.set(Calendar.DAY_OF_MONTH, +10);
            String personalToken = ws.auth.createPersonalAccessToken("openkm", expiration);
            log.info(personalToken);
        } catch (Exception e) {
            log.error(e.getMessage());
        }
    }
}
```

## Bookmark samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Bookmark methods from "**bookmark**" class as is shown below:

```
ws.bookmark.getUserBookmarks()
```

### Methods

#### getUserBookmarks

Description:

Method	Return values	Description
<b>getUserBookmarks()</b>	<b>List&lt;Bookmark&gt;</b>	Returns a list of all the bookmarks of a user.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Bookmark;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        for (Bookmark bookmark : ws.bookmark.getUserBookmarks()) {
            System.out.println(bookmark);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**create**

Description:

Method	Return values	Description
<b>create(String uuid, String name)</b>	<b>void</b>	Create a new bookmark.
The value of the <b>uuid</b> is the UUID of the node ( document, folder, mail or record ).		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.bookmark.create("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "test bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**rename**

Description:

Method	Return values	Description
<b>rename(int bookmarkId, String name)</b>	<b>void</b>	Rename a bookmark

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int bookmarkId = 1;
            ws.bookmark.rename(bookmarkId, "set bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## delete

Description:

Method	Return values	Description
<b>delete(int bookmarkId)</b>	<b>void</b>	Delete a bookmark.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int bookmarkId = 1;
            ws.bookmark.delete(bookmarkId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}  
}
```

## Conversion samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Conversion methods from "**conversion**" class as is shown below:

```
ws.conversion.doc2pdf(is, "test.docx");
```

### Methods

#### doc2pdf

Description:

Method	Return values	Description
<b>doc2pdf(InputStream is, String fileName)</b>	<b>InputStream</b>	Retrieve the uploaded document converted to PDF format.
<p>The parameter fileName is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.</p>		
<p> The openoffice service must be enabled in OpenKM server to get it running.</p>		

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.docx");
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.pdf");
            InputStream convertedStream = ws.conversion.doc2pdf(is, "test.docx");
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```


**imageConvert**

Description:

Method	Return values	Description
<b>imageConvert(InputStream is, String fileName, String params, String dstMimeType)</b>	<b>InputStream</b>	Retrieve the uploaded image with transformation.

The variable fileName is the document file name. Application uses this variable to identify by document extension the document MIME TYPE.

The parameter dstMimeType is the expected document MIME TYPE result.

 Using this method you are really executing on server side the ImageMagick convert tool.

You can set a lot of parameters - transformations - in **params** variable. Take a look at [ImageMagick convert tool](#) to get a complete list of them.



The image convert tool must be enabled in OpenKM server to get it running.

When params value is not empty always must contains ends with the chain "\${fileIn} \${fileOut}".

Ensure there is only a white space as separator between two parameters.

When building your integrations, we suggest installing [ImageMagic](#) software locally, and check your image transformations first from your command line.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.jpg");
            InputStream convertedStream = ws.conversion.imageConvert(is, "test.png", "-resize 50% ${fileIn} ${fileOut}");
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### html2pdf

Description:

Method	Return values	Description
<b>html2pdf(String url)</b>	<b>InputStream</b>	Retrieve the PDF of an URL.



The HTML to PDF conversion tool must be enabled in OpenKM server to get it running.

Example:

```

package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.pdf");
            InputStream is = ws.conversion.html2pdf("https://www.openkm.com/es/");
            IOUtils.copy(is, fos);
            is.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**doc2txt**

Description:

Method	Return values	Description
<b>doc2txt(InputStream is, String fileName)</b>	<b>String</b>	Extracts the text from the upload document.



Must be enabled a Text Extractor for the document MIME TYPE in OpenKM server to get it running.

Example:

```

package com.openkm;

import java.io.FileInputStream;
    
```

```
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.docx");
            System.out.println(ws.conversion.doc2txt(is, "test.docx"));
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**img2txt**

Description:

Method	Return values	Description
<b>img2txt(InputStream is, String fileName)</b>	<b>String</b>	Extracts the text from the uploaded image.

 The OCR engine must be enabled in OpenKM server to get it running.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            System.out.println(ws.conversion.img2txt(is, "test.png"));
        }
    }
}
```

```


        is.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**barcode2txt**

Description:

Method	Return values	Description
<b>barcode2txt(InputStream is, String fileName)</b>	<b>String</b>	Extracts the barcode text from the uploaded image.



The Bar Code tool must be enabled in OpenKM server to get it running.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            System.out.println(ws.conversion.barcode2txt(is, "test.png"));
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## Dashboard samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method



At this point you can use all the Dashboard methods from "**dashboard**" class as is shown below:

```
ws.dashboard.getUserCheckedOutDocuments(0, 5);
```

### Methods

#### getUserCheckedOutDocuments

Description:

Method	Return values	Description
<b>getUserCheckedOutDocuments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the documents in edition by the user.
<div style="background-color: #e0f0e0; padding: 10px; border: 1px dashed #ccc;"> <p> The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.</p> <ul style="list-style-type: none"> <li>• The parameter "limit" is used to limit the number of results returned.</li> <li>• The parameter "offset" says to skip that many results before the beginning to return results.</li> </ul> </div>		
<div style="background-color: #e0f0ff; padding: 10px; border: 1px dashed #ccc;"> <p> For example if your query has 1000 results, but you only want to return the first 10, you should use these values:</p> </div>		

- 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.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserCheckedOutDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUserLastModifiedDocuments**

Description:

Method	Return values	Description
<b>getUserLastModifiedDocuments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last documents modified by the user.

 The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

**i** 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastModifiedDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getUserLockedDocuments**

Description:

Method	Return values	Description
<b>getUserLockedDocuments(int offset, int</b>	<b>DashboardResultSet</b>	Returns a list of the documents locked by the

**limit)**

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


**getUserLockedRecords**

Description:

Method	Return values	Description
<b>getUserLockedRecords(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the records 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedRecords(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

```
}
}
```


### getUserLockedFolders

Description:

Method	Return values	Description
<b>getUserLockedFolders(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the folders locked by the user.

 The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

 For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedFolders(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
```

```


        System.out.println(dashboardResult.getNode());
    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```


**getUserLockedMails**

Description:

Method	Return values	Description
<b>getUserLockedMails(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the mails locked by the user.

 The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

 For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    DashboardResultSet resultSet = ws.dashboard.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
<b>getUserSubscribedDocuments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the documents subscribed 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

```

```
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);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


**getUserSubscribedFolders**

Description:

Method	Return values	Description
<b>getUserSubscribedFolders(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the folders subscribed 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedFolders(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**getUserSubscribedRecords**

Description:

Method	Return values	Description
<b>getUserSubscribedRecords(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the records subscribed 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedRecords(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUserLastCreatedDocuments**

Description:

Method	Return values	Description
<b>getUserLastCreatedDocuments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last documents created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getUserLastDocumentsNotesCreated**

Description:

Method	Return values	Description
<b>getUserLastDocumentsNotesCreated(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last documents notes created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastDocumentsNotesCreated(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getUserLastCreatedFolders**

Description:

Method	Return values	Description
<b>getUserLastCreatedFolders(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last folders created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedFolders(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUserLastFoldersNotesCreated**

Description:

Method	Return values	Description
--------	---------------	-------------

<b>getUserLastFoldersNotesCreated(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last folders notes created by the user.
<p> The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.</p> <ul style="list-style-type: none"> <li>• The parameter "limit" is used to limit the number of results returned.</li> <li>• The parameter "offset" says to skip that many results before the beginning to return results.</li> </ul>		
<p> For example if your query has 1000 results, but you only want to return the first 10, you should use these values:</p> <ul style="list-style-type: none"> <li>• limit=10</li> <li>• offset=0</li> </ul> <p>Now suppose you want to show the results from 11-20, you should use these values:</p> <ul style="list-style-type: none"> <li>• limit=10</li> <li>• offset=10</li> </ul>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastFoldersNotesCreated(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**getUserLastCreatedRecords**

Description:

Method	Return values	Description
<b>getUserLastCreatedRecords(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last records created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedRecords(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        }
    }
}
    
```

```


    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```


**getUserLastRecordsNotesCreated**

Description:

Method	Return values	Description
<b>getUserLastRecordsNotesCreated(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last records notes created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    DashboardResultSet resultSet = ws.dashboard.getUserLastRecordsNotesCreated(0, 5);
    System.out.println("Total: " + resultSet.getTotal());
    for (DashboardResult dashboardResult : resultSet.getResults()) {
        System.out.println(dashboardResult.getNode());
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```


**getUserLastDownloadedDocuments**

Description:

Method	Return values	Description
<b>getUserLastDownloadedDocuments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last documents downloaded 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        DashboardResultSet resultSet = ws.dashboard.getUserLastDownloadedDocuments(0, 5);
        System.out.println("Total: " + resultSet.getTotal());
        for (DashboardResult dashboardResult : resultSet.getResults()) {
            System.out.println(dashboardResult.getNode());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```


### getUserLastImportedMails

Description:

Method	Return values	Description
<b>getUserLastImportedMails(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last mails imported 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMails(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**getUserLastMailsNotesCreated**

Description:

Method	Return values	Description
<b>getUserLastMailsNotesCreated(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last mails notes created 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastMailsNotesCreated(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getUserLastImportedMailAttachments**


Description:

Method	Return values	Description
<b>getUserLastImportedMailAttachments(int offset, int limit)</b>	<b>DashboardResultSet</b>	Returns a list of the last mails imported with attachments by the user.



The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.



For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMailAttachments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getUserSearches**

Description:

Method	Return values	Description
<b>getUserSearches()</b>	<b>List&lt;QueryParams&gt;</b>	Returns a list of the searches saved by the user as user news.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        List<QueryParams> results = ws.dashboard.getUserSearches();
        for (QueryParams userSearch : results) {
            System.out.println(userSearch);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

**findUserSearches**

Description:

Method	Return values	Description
<b>findUserSearches(long qpId)</b>	<b>List&lt;DashboardNodeResult&gt;</b>	Returns a list of nodes based in a search saved by the user ( search of type user news ).

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardNodeResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long qpId = 1;
            List<DashboardNodeResult> results = ws.dashboard.findUserSearches(qpId);
            for (DashboardNodeResult nodeResult : results) {
                System.out.println(nodeResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## Document samples

### Basics

 Example of uuid:

- Using UUID -> `"f123a950-0329-4d62-8328-0ff500fd42db"`;


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method **"login"**. You can access the **"login"** method from webservice object **"ws"** as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Document methods from **"document"** class as is shown below:

```
ws.document.create("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is);
```

### Methods

#### create

Description:

Method	Return values	Description
<b>create(String uuid, String name, InputStream is)</b>	<b>Document</b>	Creates a new document. Return an object Document with the properties of the created document.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/okm/logo.png");
            Document doc = ws.document.create("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is);
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**create**

Description:

Method	Return values	Description
<b>create(String uuid, String name, InputStream is, long nodeClass)</b>	<b>Document</b>	Creates a new document with nodeClass. Return an object Document with the properties of the created document.
<p>The values of the uuid parameter should be a folder or record node UUID.</p> <p>The nodeClass parameter should be a valid bussiness type ( serie ) value. A nodeClass with value 0 means there's no business type ( serie ) selected.</p>		

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

```

```
import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/okm/logo.png");
            long nodeClass = 0;
            Document doc = ws.document.create("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is, nodeClass);
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**create**

Description:

Method	Return values	Description
<b>create(String uuid, File file)</b>	<b>Document</b>	Creates a new document and returns as a result an object Document with the properties of the created document.
The values of the <b>uuid</b> parameter should be a folder or record node UUID.		

Example:

```
package com.openkm;

import java.io.File;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    File file = new File("/home/openkm/okm/logo.png");
    Document doc = ws.document.create("151f3a54-f370-47d6-801a-d20faecec180", file);
    System.out.println(doc);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**delete**

Description:

Method	Return values	Description
<b>delete(String uuid)</b>	<b>void</b>	Delete a document.



When a document is deleted it is automatically moved to /okm:trash/userId folder.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.delete("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProperties**

Description:

Method	Return values	Description
<b>getProperties(String uuid)</b>	<b>Document</b>	Return the document properties.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.getProperties("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getContent**


Description:

Method	Return values	Description
<b>getContent(String uuid)</b>	<b>InputStream</b>	Retrieve document content - binary data - of the actual document version.



In case you sent the file across a Servlet response we suggest set the content length with:

```
Document doc = ws.getDocumentProperties("1ec49da9-1746-4875-ae32-9281d7303a62");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```



We've found wrong size problems while using:

```
response.setContentLength(is.available());
```

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            OutputStream fos = new FileOutputStream("/home/openkm/logo.png");
            InputStream is = ws.document.getContent("1ec49da9-1746-4875-ae32-9281d7303a62");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getContentByVersion**


Description:

Method	Return values	Description
<b>getContentByVersion(String uuid, String versionName)</b>	<b>InputStream</b>	Retrieves a document content ( binary data ) from a specific document version.



In case you sent the file across a Servlet response we suggest set the content length with:

```
Document doc = ws.getProperties("1ec49da9-1746-4875-ae32-9281d7303a62");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```



We've found wrong size problems while using:

```
response.setContentLength(is.available());
```

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            OutputStream fos = new FileOutputStream("/home/openkm/logo.png");
            InputStream is = ws.document.getContentByVersion("/okm:root/logo.png", "1.1");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## getChildren

Description:

Method	Return values	Description
<b>getChildren(String fldId)</b>	<b>List&lt;Document&gt;</b>	Returns a list of all documents which their parent is fldId.
The parameter fldId can be a folder or a record node UUID.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

```

```

        for (Document doc : ws.document.getChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
            System.out.println(doc);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**rename**

Description:

Method	Return values	Description
<b>rename(String uuid, String newName)</b>	<b>Document</b>	Changes the name of a document.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Document doc = ws.document.rename("1ec49da9-1746-4875-ae32-9281d7303a62", "logo_rename.png");
            System.out.print(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setProperties**

Description:

Method	Return values	Description
<b>setProperties(String uuid, String title, String description, String lang, List&lt;String&gt; keywords, List&lt;String&gt; categories)</b>	<b>void</b>	Changes some document properties.

The parameter lang must be ISO 691-1 compliant.

 More information at: [https://en.wikipedia.org/wiki/List\\_of\\_ISO\\_639-1\\_codes](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> keywords = new ArrayList<>();
            keywords.add("test");
            keywords.add("invoice");

            List<String> categories = new ArrayList<>();
            categories.add("58c9b25f-d83e-4006-bd78-e26d7c6fb648");

            ws.document.setProperties("1ec49da9-1746-4875-ae32-9281d7303a62", "new title", "some description", "es");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**setLanguage**

Description:

Method	Return values	Description
<b>setLanguage(String uuid, String lang)</b>	<b>void</b>	Sets the document language.

The parameter lang must be ISO 691-1 compliant.

 More information at: [https://en.wikipedia.org/wiki/List\\_of\\_ISO\\_639-1\\_codes](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.setLanguage("1ec49da9-1746-4875-ae32-9281d7303a62", "en");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### setTitle

Description:

Method	Return values	Description
<b>setTitle(String uuid, String title)</b>	<b>void</b>	Sets the document title.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.setTitle("1ec49da9-1746-4875-ae32-9281d7303a62", "Some title here");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### checkout

Description:

Method	Return values	Description
<b>checkout(String uuid)</b>	<b>void</b>	Marks the document for edition.

Only one user can modify a document at a time.

Before starting edition you must do a checkout action that locks the edition process for other users and allows edition only to the user who has executed the action.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.checkout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point the document is locked for other users except for the user who executed the action
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**cancelCheckout**

Description:

Method	Return values	Description
<b>cancelCheckout(String uuid)</b>	<b>void</b>	Cancels a document edition.



This action can only be done by the user who previously executed the checkout action.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // At this point the document is locked for other users except for the user who executed the action
            ws.document.cancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**forceCancelCheckout**

Description:

Method	Return values	Description
<b>forceCancelCheckout(String uuid)</b>	<b>void</b>	Cancels a document edition.

This method allows to cancel edition on any document.

It is not mandatory to execute this action by the same user who previously executed the checkout action.

 This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
// At this point the document is locked for other users except for the user who executed the action
ws.document.forceCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
// At this point other users are allowed to execute a checkout and modify the document
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**isCheckedOut**

Description:

Method	Return values	Description
<b>isCheckedOut(String docId)</b>	<b>Boolean</b>	Returns a boolean that indicates if the document is on edition or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("Is the document checkout:" + ws.document.isCheckedOut("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**checkin**

Description:

Method	Return values	Description
<b>checkin(String uuid, InputStream is, String comment)</b>	<b>Version</b>	Updates a document with a new version and returns an object with new Version values.

 Only the user who started the edition - checkout - is allowed to update the document.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.checkin("1ec49da9-1746-4875-ae32-9281d7303a62", is, "optional some comment");
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**checkin**

Description:

Method	Return values	Description
<b>checkin(String docId, InputStream is, String comment, int increment)</b>	<b>Version</b>	Updates a document with a new version and returns an object with new Version values.

 The value of increment variable must be 1 or greater.  
The valid values of increment variable depends on the VersionNumberAdapter you have enabled.

 Only the user who started the edition - checkout - is allowed to update the document.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.checkin("1ec49da9-1746-4875-ae32-9281d7303a62", is, "optional some comment", 2);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

**purge**

Description:

Method	Return values	Description
<b>purge(String uuid)</b>	<b>void</b>	The document is definitely removed from repository.

Usually you will purge documents into /okm:trash/userId - the personal trash user locations - but is possible to directly purge any document from the whole repository.



When a document is purged only will be able to be restored from a previously repository backup. The purge action removes the document definitely from the repository.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        ws.document.purge("1ec49da9-1746-4875-ae32-9281d7303a62");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**move**

Description:

Method	Return values	Description
<b>move(String uuid, String dstId)</b>	<b>void</b>	Move document into some folder or record.
The values of the dstId parameter should be a folder or record UUID.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.move("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab7-ae61-4628-8010-1103d6950c63");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**copy**


Description:

Method	Return values	Description

<b>copy(String uuid, String dstId, String newName)</b>	<b>void</b>	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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.copyDocument("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab7-ae61-4628-8010-1103");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getVersionHistorySize**

Description:

Method	Return values	Description
<b>getVersionHistorySize(String uuid)</b>	<b>long</b>	Returns the sum in bytes of all documents into documents history.

Example:

```

package com.openkm;

import java.util.Locale;
    
```

```

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String[] UNITS = new String[]{"B", "KB", "MB", "GB", "TB", "PB", "EB"};
            long bytes = ws.document.getVersionHistorySize("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e");
            String value = "";

            for (int i = 6; i > 0; i--) {
                double step = Math.pow(1024, i);
                if (bytes > step) {
                    value = String.format(Locale.ROOT, "%3.1f %s", bytes / step, UNITS[i]);
                }
            }

            if (value.isEmpty()) {
                value = Long.toString(bytes) + " " + UNITS[0];
            }

            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## isValid

Description:

Method	Return values	Description
<b>isValid(String docId)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is a document or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

```

```

        System.out.println(ws.document.isValid("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getPath**

Description:

Method	Return values	Description
<b>getPath(String uuid)</b>	<b>String</b>	Converts a document UUID to document path.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.getPath("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getDetectedLanguages**

Description:

Method	Return values	Description
<b>getDetectedLanguages()</b>	<b>List&lt;String&gt;</b>	Return a list of available document languages what OpenKM can identify.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;

```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (String lang : ws.document.getDetectedLanguages()) {
                System.out.println(lang);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**extendedCopy**

Description:

Method	Return values	Description
<b>extendedCopy(String nodeId, String dstId, String name, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)</b>	<b>void</b>	Copies a document with associated data into some folder or record.

The values of the nodeId parameter should be a Document UUID.

The values of the dstId parameter should be a folder or a record UUID.

When the parameter newName value is null, the document will preserve the same name.

**i** By default only the binary data and the security grants, the metadata, keywords, etc. of the document are not copied.

Additional:

- When the category parameter is true the original values of the categories will be copied.
- When the keywords parameter is true the original values of the keywords will be copied.
- When the propertyGroups parameter is true the original values of the metadata groups will be copied.
- When the notes parameter is true the original value of the notes will be copied.
- When the security parameter is true the original value of the security will be copied.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Document document = ws.document.extendedCopy("055b5206-35d0-4351-ba92-c304bbba7ddf", "2dcccdee1
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getExtractedText**

Description:

Method	Return values	Description
<b>getExtractedText(String uuid)</b>	<b>String</b>	Returns an String with the extracted text by text extractor process.

When a document is uploaded to OpenKM, it goes into text extraction queue. There's a crontab tab that periodically process this queue and extract document contents.



Unfortunately there is not a direct way to know if a document has been processed or not from the API, because this information is only stored at database level.

Although this restriction, from API - only administrators users - can do database queries to retrieve this information. Check [Repository samples](#) for accessing database level.



More information at [Statistics](#).

Example:

```

package com.openkm;
    
```

```

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.io.FileInputStream;
import java.io.InputStream;

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);

            InputStream is = new FileInputStream("/home/openkm/test.pdf");
            ws.document.setExtractedText("301de803-f4ae-48f1-8505-e83b26716956", is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setExtractedText**

Description:

Method	Return values	Description
<b>setExtractedText(String uuid, InputStream is)</b>	<b>void</b>	Set extracted text

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String text = ws.document.getExtractedText("46762f90-82c6-4886-8d21-ad3017dd78a7");
            System.out.println(text);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**getThumbnail**

Description:

Method	Return values	Description
<b>getThumbnail(String uuid, ThumbnailType type)</b>	<b>InputStream</b>	Returns thumbnail image data.

Available types:

- ThumbnailType.THUMBNAIL\_PROPERTIES ( shown in properties view )
- ThumbnailType.THUMBNAIL\_LIGHTBOX ( shown in light box )
- ThumbnailType.THUMBNAIL\_SEARCH ( shown in search view )

 Each thumbnail type has its own image dimensions.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ThumbnailType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            OutputStream fos = new FileOutputStream("/home/openkm/invoice.png");
            InputStream is = ws.document.getThumbnail("46762f90-82c6-4886-8d21-ad3017dd78a7", ThumbnailType.T
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### createFromTemplate


Description:


Method	Return values	Description
<b>createFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map&lt;String, String&gt; properties)</b>	<b>Document</b>	Creates a new document from the template and returns an object Document.

The **uuid** parameter is the UUID value of the template file.

The **dstPath** value is the document destination path.

When the template uses metadata groups to fill in fields, then these values are mandatory and must be set into properties parameter.

 For more information about Templates and metadata check: [Creating templates](#) [Creating templates](#)

 Additional:

- When category parameter is true the original values of the categories will be copied.
- When keywords parameter is true the original values of the keywords will be copied.
- When notes parameter is true the original values of the notes will be copied.

Example:

 The example below is based on [Creating PDF template](#) sample.

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
    }
}
    
```

```

OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

try {
    ws.login(username, password);

    Map<String, String> properties = new HashMap<>();
    // okg:tpl
    properties.put("okp:tpl.name", "sdk name");
    Calendar cal = Calendar.getInstance();
    // Value must be converted to String ISO 8601 compliant
    properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
    properties.put("okp:tpl.language", "[ \"java\" ]");

    // Property okg:technology
    properties.put("okp:technology.comment", "sdk name");

    Document document = ws.document.createFromTemplate("5e72dec8-2409-405b-ac15-d964feb3d7bb", "/ok
    System.out.println(document);
} catch (Exception e) {
    e.printStackTrace();
}
}
}
}

```

### updateFromTemplate


Description:

Method	Return values	Description
<b>updateFromTemplate(String uuid, String dstId, Map&lt;String, String&gt; properties)</b>	<b>Document</b>	Updates a document previously created from the template and returns an object Document.

The uuid value is the UUID value of the template file.

The dstId is the document to updated UUID.

This method only has sense when the template uses metadata groups to fill in fields.

 For more information about Templates and metadata take a look at [Creating templates](#) .

Example:

 The example below is based on [Creating PDF template](#) sample.

```

package com.openkm;
import java.util.Calendar;

```

```
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

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);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:tpl.name", "update Some name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
            properties.put("okp:tpl.language", "[ \"python\" ]");

            ws.document.updateFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "058b9379-b441-454d-8590");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getAnnotations**

Description:

Method	Return values	Description
<b>getAnnotations(String uuid, String versionName)</b>	<b>String</b>	Return the document annotations of some document version.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.getAnnotations("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "1.0"));
        }
    }
}
```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getDifferences**

Description:

Method	Return values	Description
<b>getDifferences(String uuid, String versionName1, String versionName2)</b>	<b>InputStream</b>	Return a PDF document with the differences between two document versions.

Example:

```

package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = ws.document.getDifferences("46762f90-82c6-4886-8d21-ad3017dd78a7", "1.0", "1.1");
            FileOutputStream fos = new FileOutputStream("/home/openkm/okm/out.pdf");
            IOUtils.copy(is, fos);
            is.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getCheckedOut**

Description:

Method	Return values	Description

<b>getCheckedOut()</b>	<b>List&lt;Document&gt;</b>	Return a list of documents checkout by the user.
------------------------	-----------------------------	--

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Document doc : ws.document.getCheckedOut()) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**setNodeClass**

Description:

Method	Return values	Description
<b>setDocumentNodeClass(String uuid, long ncId)</b>	<b>void</b>	Set the NodeClass.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long ncId = 2;
            ws.document.setNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        }
    }
}
    
```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**setDispositionStage**

Description:

Method	Return values	Description
<b>setDispositionStage(String uuid, long stage)</b>	<b>void</b>	Set the disposition stage

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stage = 1;
            ws.document.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setDescription**

Description:

Method	Return values	Description
<b>setDescription(String uuid, String description)</b>	<b>void</b>	Set a description.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

```

```

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);
            ws.document.setDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**createWizard**

Description:

Method	Return values	Description
<b>createWizard(String uuid, String name, long nodeClass, InputStream is)</b>	<b>WizardNode</b>	Create a new document with wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

**i** The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        InputStream is = new FileInputStream("/home/openkm/okm/logo.png");
        WizardNode wn = ws.document.createWizard("1f323e88-64ee-4f57-91e2-9276f8c603f9", "logo.png", 0, is);
        System.out.print(wn);
        IOUtils.closeQuietly(is);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**isOCRDataCaptureSupported**

Description:

Method	Return values	Description
<b>isOCRDataCaptureSupported(String uuid)</b>	<b>boolean</b>	Check if the node supports an OCR data capture.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

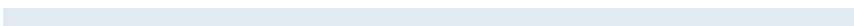
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            if (ws.document.isOCRDataCaptureSupported("b2f88679-e3fd-4f97-bf0e-abf76f9ec499")) {
                System.out.println("Yes supported OCR Data capture");
            } else {
                System.out.println("No supported OCR Data capture");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**recognize**

Description:



Method	Return values	Description
<b>recognize(String uuid)</b>	<b>OCRRecognise</b>	Return an OCRRecognise object.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.OCRRecognise;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            OCRRecognise ocr = ws.document.recognize("b2f88679-e3fd-4f97-bf0e-abf76f9ec499");
            System.out.println(ocr);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**captureData**

Description:

Method	Return values	Description
<b>captureData(String uuid, long templateId)</b>	<b>void</b>	Proceed to the OCR data capture using OCR template identified by templateId.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long templateId = 1;
            ws.document.captureData("f123a950-0329-4d62-8328-0ff500fd42db", templateId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getNumberOfPages**

Description:

Method	Return values	Description
<b>getNumberOfPages(String uuid)</b>	<b>int</b>	Get the number of pages of a document.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("The number of pages is : " + ws.document.getNumberOfPages("b2f88679-e3fd-4f97-bf0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getPageAsImage**

Description:

Method	Return values	Description
<b>getPageAsImage(String uuid, int pageNumber, int maxWidth, int maxHeight)</b>	<b>String</b>	Return specific page as an image encoded as a String in b64.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.getPageAsImage("3fe350e2-69e8-4681-a97c-6ac4ba6c1524", 1, 150, 150))
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getLiveEditRestrictedMimeTypes**

Description:

Method	Return values	Description
<b>getLiveEditRestrictedMimeTypes()</b>	<b>List&lt;String&gt;</b>	Return a list of mime types what can not be used with liveedit feature.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> mimeTypes = ws.document.getLiveEditRestrictedMimeTypes();
            for (String mimeType : mimeTypes) {
                System.out.println(mimeType);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

```


    }
}

```

### liveEditCheckin

Description:

Method	Return values	Description
<b>liveEditCheckin(String uuid, String comment, int increment)</b>	<b>void</b>	It does a live edit checkin.

 The method should be used only when the document has been edited by live edit.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.liveEditCheckin("8b9a32c8-db65-41c8-a2a0-af03761f60b6", "comment 1.1", 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### liveEditCancelCheckout

Description:

Method	Return values	Description
<b>liveEditcancelCheckout(String uuid)</b>	<b>void</b>	Cancels a document live edition.

 This action can only be done by the user who previously executed the checkout action.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // At this point the document is locked for other users except for the user who executed the action
            ws.document.liveEditCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### liveEditForceCancelCheckout

Description:

Method	Return values	Description
<b>liveEditForceCancelCheckout(String uuid)</b>	<b>void</b>	Cancels a document live edition.

This method allows to cancel live edition on any document.

It is not mandatory to execute this action by the same user who previously executed the checkout action.

This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    // At this point the document is locked for other users except for the user who executed the action
    ws.document.liveEditForceCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
    // At this point other users are allowed to execute a checkout and modify the document
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**mergePdf**

Description:

Method	Return values	Description
<b>mergePdf(String destinationUuid, String docName, List&lt;String&gt; uuids)</b>	<b>String</b>	Merge two or more PDF files.
<p>The parameters <b>destinationUuid</b> should be any valid folder or record UUID.</p>		

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> uuids = new ArrayList<>();
            uuids.add("3fe350e2-69e8-4681-a97c-6ac4ba6c1524");
            uuids.add("8b9a32c8-db65-41c8-a2a0-af03761f60b6");
            String fldUuid = "7213e597-c147-46c9-a90b-ac3966b3114b";
            String uuid = ws.document.mergePdf(fldUuid, "MergePdf.pdf", uuids);
            Document pdfDoc = ws.document.getDocumentProperties(uuid);
            System.out.println(pdfDoc.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**liveEditSetContent**

Description:

Method	Return values	Description
<b>liveEditSetContent(String uuid, InputStream is)</b>	<b>Void</b>	Update the content of the document edited with live edit feature.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.liveEditSetContent("1ec49da9-1746-4875-ae32-9281d7303a62", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**isAttachment**

Description:

Method	Return values	Description
<b>isAttachment(String docId)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is a mail attachment or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.isAttachment("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**isConvertibleToPDF**

Description:

Method	Return values	Description
<b>isConvertibleToPDF(String docId)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is convertible to PDF or not.


In case of a node of type **PDF file** will be returned a **false** value.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.document.isConvertibleToPDF("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        e.printStackTrace();
    }
}
}

```

**getPdf**

Description:

Method	Return values	Description
<b>getPdf(String uuid)</b>	<b>InputStream</b>	Returns a PDF of the document.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Document doc = ws.document.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            InputStream is = ws.document.getPdf(doc.getUuid());
            OutputStream fos = new FileOutputStream("/home/openkm/book.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**saveAsPdf**

Description:

Method	Return values	Description

<b>saveAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)</b>	<b>Document</b>	Save the document as PDF in the OpenKM repository.
---	-----------------	--

The value of the **uuid** parameter should be a Document UUID.

When the parameter **newName** value is null, the document will preserve the same name.

**i** Additional:

- When the **propertyGroups** parameter is true the metadata groups of the source are copied to the target.
- When the **security** parameter is true the security of the source is copied to the target.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Document doc = ws.document.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            Document docPdf = ws.document.saveAsPdf(doc.getUuid(), "book.pdf", true, true);
            System.out.println("Document pdf: " + docPdf.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**webPageImport**

Description:

Method	Return values	Description
<b>webPageImport(String uuid, String url)</b>	<b>void</b>	Save a document as PDF with the content of the web pag.

The value of the **uuid** parameter should be a folder or record node UUID.

The value of the **url** parameter is any web page

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.webPageImport("271cc3aa-218e-4574-8065-c34c704f4a20", "https://www.google.com/");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### setIndexable

Description:

Method	Return values	Description
<b>setIndexable(String uuid, boolean enabled)</b>	<b>void</b>	Sets the document indexable.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.document.setIndexable("19938640-fd34-421c-90c8-6b435e4b7d79", true);
            Document document = ws.document.getDocumentProperties("19938640-fd34-421c-90c8-6b435e4b7d79");
        }
    }
}
```

```

        System.out.println("Document indexable:" + document.isIndexable());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getXRefs**

Description:

Method	Return values	Description
<b>getXRefs(String uuid)</b>	<b>List&lt;XRef&gt;</b>	Returns a list of the document references of a Autocad document

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.XRef;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            for (XRef xRef : ws.document.getXRefs("5ce6dd37-7472-404a-878e-274005a2d6db")) {
                System.out.println(xRef);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**refresh**

Description:

Method	Return values	Description
<b>refresh(String uuid)</b>	<b>void</b>	refresh references of an Autocad document

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            ws.document.refresh("5ce6dd37-7472-404a-878e-274005a2d6db");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## FilePlan samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the FilePlan methods from "**filePlan**" class as is shown below:

```
ws.filePlan.getRootSections();
```

### Methods

#### getRootSections

Description:

Method	Return values	Description
<b>getRootSections()</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    List<NodeClassSectionDefinition> list = ws.filePlan.getRootSections();
    for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
        System.out.println(nodeClassSectionDefinition);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

### getRootSectionsFilteredBySecurity

Description:

Method	Return values	Description
<b>getRootSectionsFilteredBySecurity(int permissions)</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition by permissions.

**i** The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<NodeClassSectionDefinition> list = ws.filePlan.getRootSectionsFilteredBySecurity(Permission.ALL_GR);
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

        e.printStackTrace();
    }
}

```

**getChildrenSections**

Description:

Method	Return values	Description
<b>getChildrenSections(long parentId)</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition by parent section.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int parentId = 1;
            List<NodeClassSectionDefinition> list = ws.filePlan.getChildrenSections(parentId);
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getChildrenSectionsFilteredBySecurity**

Description:

Method	Return values	Description
<b>getChildrenSectionsFilteredBySecurity(long parentId)</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition

<b>parentId, int permissions)</b>	by parent section and permissions.
-----------------------------------	------------------------------------

**i** The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int parentId = 4;
            List<NodeClassSectionDefinition> list = ws.filePlan.getChildrenSectionsFilteredBySecurity(parentId, Permission.READ);
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getChildrenClasses**

Description:

Method	Return values	Description
<b>getChildrenClasses(long sectionId)</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass by section.

Example:

```

package com.openkm;

import java.util.List;
    
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int sectionId = 3;
            List<NodeClass> list = ws.filePlan.getChildrenClasses(sectionId);
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getChildrenClassesFilteredBySecurity**

Description:

Method	Return values	Description
<b>getChildrenClassesFilteredBySecurity(long sectionId, int permissions)</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass by section and permissions.

**i** The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    int sectionId = 3;
    List<NodeClass> list = ws.filePlan.getChildrenClassesFilteredBySecurity(sectionId, Permission.WRITE);
    for (NodeClass nodeClass : list) {
        System.out.println(nodeClass);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**findNodeClassByPk**

Description:

Method	Return values	Description
<b>findNodeClassByPk(long id)</b>	<b>NodeClass</b>	Return a NodeClass.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int id = 4;
            NodeClass nodeClass = ws.filePlan.findNodeClassByPk(id);
            System.out.println(nodeClass);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**findElectronicRecordClasses**

Description:

Method	Return values	Description
<b>findElectronicRecordClasses()</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass what are Electronic Records.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<NodeClass> list = ws.filePlan.findElectronicRecordClasses();
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**findElectronicRecordClassesFilteredBySecurity**

Description:

Method	Return values	Description
<b>findElectronicRecordClassesFilteredBySecurity(int permissions)</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass by section and permissions what are Electronic Records.

 The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
    
```

```
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {


    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClass> list = ws.filePlan.findElectronicRecordClassesFilteredBySecurity(Permission.ALL_GRANTED);
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**findFilteredByCodeOrNameFilteredBySecurity**

Description:

Method	Return values	Description
<b>findFilteredByCodeOrNameFilteredBySecurity(String code, String name, int permissions)</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass by code, name and permissions.

 The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
```

```

ws.login(user, password);
List<NodeClass> list = ws.filePlan.findFilteredByCodeOrNameFilteredBySecurity("1.", "", Permission.ALL_G
for (NodeClass nodeClass : list) {
    System.out.println(nodeClass);
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**findSectionFiltered**

Description:

Method	Return values	Description
<b>findSectionFiltered(String code, String name, int permissions)</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition by code, name and permissions.

**i** The class **com.openkm.sdk4j.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClassSectionDefinition> list = ws.filePlan.findSectionFiltered("1.", "invoice", Permission.ALL_GRAI
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getNodeClassBreadcrumb

Description:

Method	Return values	Description
<b>getNodeClassBreadcrumb(long sectionId)</b>	<b>List&lt;NodeClassSectionDefinition&gt;</b>	Returns a list of the NodeClassSectionDefinition what are Electronic Records.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int sectionId = 3;
            List<NodeClassSectionDefinition> list = ws.filePlan.getNodeClassBreadcrumb(sectionId);
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### findAllNodeClasses

Description:

Method	Return values	Description
<b>findAllNodeClasses()</b>	<b>List&lt;NodeClass&gt;</b>	Returns a list of the NodeClass.

Example:

```

package com.openkm;
    
```

```
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<NodeClass> list = ws.filePlan.findAllNodeClasses();
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getRootSectionsIdWithChildren**

Description:

Method	Return values	Description
getRootSectionsIdWithChildren()	<b>List&lt;Long&gt;</b>	Returns a list of the section ids what have childs and parent is root.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);
            List<Long> list = ws.filePlan.getRootSectionsIdWithChildren();
            for (long id : list) {
                System.out.println(id);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
  }
}

```

### getChildrenSectionsIdByTenantWithChildren

Description:

Method	Return values	Description
getChildrenSectionsIdByTenantWithChildren(long parentId)	<b>List&lt;Long&gt;</b>	Returns a list of the section ids what have childs with parent equals to parentId and filtered by tenant.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);
            long parentId = 1;
            List<Long> list = ws.filePlan.getChildrenSectionsIdByTenantWithChildren(parentId);
            for (long id : list) {
                System.out.println(id);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## Folder samples

### Basics



Example of uuid:

- Using UUID -> "f123a950-0329-4d62-8328-0ff500fd42db";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "login". You can access the "login" method from webservice object "ws" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Folder methods from "folder" class as is shown below:

```
ws.folder.create("1be884f4-5758-4985-94d1-f18bfe004db8", "test");
```

### Methods

#### create

Description:

Method	Return values	Description
<code>create(String uuid, String name)</code>	<b>Folder</b>	Creates a new folder and returns as a result an object Folder.
The parameters <b>uuid</b> should be any valid folder or record <b>UUID</b> .		

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Folder folder = ws.folder.create("1be884f4-5758-4985-94d1-f18bfe004db8", "test");
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getProperties**

Description:

Method	Return values	Description
<b>getProperties(String uuid)</b>	<b>Folder</b>	Return the folder properties.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.folder.getProperties("76f4155e-42af-4be4-9a1c-756497616fb6"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**delete**

Description:

Method	Return values	Description
<b>delete(String fldId)</b>	<b>void</b>	Delete a folder.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.delete("76f4155e-42af-4be4-9a1c-756497616fb6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**rename**

Description:

Method	Return values	Description
<b>rename(String fldId, String newName)</b>	<b>void</b>	Rename a folder.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Exists folder /okm:root/test
            ws.folder.rename("91264771-02b9-471d-b7a3-ba8cc49a10dd", "renamedFolder");
        }
    }
}
    
```

```

        // Folder has renamed to /okm:root/renamedFolder
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**move**

Description:

Method	Return values	Description
<b>move(String uuid, String dstId)</b>	<b>void</b>	Moves folder into some folder or record.
The values of the dstId parameter should be a folder or record UUID.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Exists folder /okm:root/test
            ws.folder.move("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8010-1103d6950c63");
            // Folder has moved to /okm:root/tmp/test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getChildren**

Description:

Method	Return values	Description
<b>getChildren(String uuid)</b>	<b>List&lt;Folder&gt;</b>	Returns a list of all folders which their parent is fldId.
The parameter uuid can be a folder or a record node.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Folder fld : ws.folder.getChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### isValid

Description:

Method	Return values	Description
<b>isValid(String uuid)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is a folder or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Return false
            ws.folder.isValid("8cd1e072-8595-4dd3-b121-41d622c43f08");

            // Return true
            ws.folder.isValid("1be884f4-5758-4985-94d1-f18bfe004db8");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
  }
}

```

**getPath**

Description:

Method	Return values	Description
<b>getPath(String uuid)</b>	<b>String</b>	Convert folder UUID to folder path.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.folder.getPath("1be884f4-5758-4985-94d1-f18bfe004db8"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**copy**

Description:

Method	Return values	Description
<b>copy(String uuid, String dstId, String newName)</b>	<b>void</b>	Copies a folder into a folder or record.

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, folder will preservate the same name.



Only the security grants are copied to destination, the metadata, keywords, etc. of the folder are not copied.  
See "**extendedCopy**" method for this feature.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.copy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8010-1103d6950c63", "n
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**extendedCopy**

Description:

Method	Return values	Description
<b>extendedCopy(String uuid, String dstId, String newName, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)</b>	<b>void</b>	Copies a folder with the associated data into some folder or record.

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, folder will preservate the same name.

**i** By default only the binary data and the security grants, the metadata, keywords, etc. of the folder 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 security parameter is true the original value of the security will be copied.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Folder folder = ws.folder.extendedCopy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-
                "new name folder", true, true, true, true);
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getContentInfo**

Description:

Method	Return values	Description
<b>getContentInfo(String uuid)</b>	<b>ContentInfo</b>	Return and object ContentInfo with information about folder.
<p>The ContentInfo object retrieves information about:</p> <ul style="list-style-type: none"> <li>• The number of folders into.</li> <li>• The number of documents into.</li> <li>• The number of records into.</li> <li>• The number of mails into.</li> <li>• The size in bytes of all objects into the folder.</li> </ul>		

Example:

```

package com.openkm;
    
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.folder.getContentInfo("ada67d44-b081-4b23-bdc1-74181cafbc5d"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**purge**

Description:

Method	Return values	Description
<b>purge(String uuid)</b>	<b>void</b>	The folder is definitely removed from the repository.

Usually you will purge folders into /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any folder from the whole repository.



When a folder is purged, it will only be able to be restored from a previously repository backup. The purge action removes the folder definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.purge("ada67d44-b081-4b23-bdc1-74181cafbc5d");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```


    }
  }
}

```

**setStyle**

Description:

Method	Return values	Description
<b>setStyle(String uuid, long styleId)</b>	<b>void</b>	Set the folder style.

 More information at [Folder style](#).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.setStyle("ada67d44-b081-4b23-bdc1-74181cafbc5d", 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**createMissingFolders**

Description:

Method	Return values	Description
<b>createMissingFolders(String fldPath)</b>	<b>String</b>	Create missing folders.

Example:

```

package com.openkm;

```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.createMissingFolders("/okm:root/missingfld1/missingfld2/missingfld3");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**setDescription**

Description:

Method	Return values	Description
<b>setDescription(String uuid, String description)</b>	<b>void</b>	Set a description.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.folder.setDescription("4c195453-246b-4ce9-86ba-b84e68d1f284", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**createFromTemplate**

Description:



Method	Return values	Description
<b>createFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map&lt;String, String&gt; properties)</b>	<b>Folder</b>	Creates a new folder from the template and returns an object Folder.

The **uuid** parameter is the UUID value of the template file.

The **dstPath** value is the folder destination path.

When the template uses metadata groups to fill in fields, then these values are mandatory and must be set into properties parameter.

**i** Additional:

- When category parameter is true the original values of the categories will be copied.
- When keywords parameter is true the original values of the keywords will be copied.
- When notes parameter is true the original values of the notes will be copied.

Example:

```

package com.openkm.example;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FolderExample {

    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);

            // createFolderFromTemplate
            Map<String, String> properties = new HashMap<>();
            // okg:tpl


```

```
properties.put("okp:tpl.name", "sdk name");
Calendar cal = Calendar.getInstance();
// Value must be converted to String ISO 8601 compliant
properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
properties.put("okp:tpl.language", "[ \"java\" ]");

// Property okg:technology
properties.put("okp:technology.comment", "sdk name");
Folder folder = ws.folder.createFromTemplate("47c70047-2924-483c-bc42-23d0cbef6b17", "/okm:root/sdk/te
System.out.println(folder);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

# Import samples

## Basics

 We suggest executing the methods below for huge import or for simple creation cases.

These methods execute fewer steps in the background either what is described in [Document samples](#) and [Folder samples](#). That means you will get extra performance in the action because there are executed less logic in the server side.


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Import methods from "**quickImport**" class as is shown below:

```
ws.quickImport.importDocument("1be884f4-5758-4985-94d1-f18bfe004db8", file);
```

## Methods

### importDocument

Description:

Method	Return values	Description
<b>importDocument(String uuid, File file)</b>	<b>String</b>	Creates a new document. Return an the UUID of the Document.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```

package com.openkm;

import java.io.File;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            File file = new File("/home/gnujavasergio/okm/logo.png");
            String uuid = ws.quickImport.importDocument("1be884f4-5758-4985-94d1-f18bfe004db8", file);
            System.out.println(uuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### importFolder

Description:

Method	Return values	Description
<b>importFolder(String uuid, String name)</b>	<b>String</b>	Creates a new folder. Return an the UUID of the folder.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String uuid = ws.quickImport.importFolder("1be884f4-5758-4985-94d1-f18bfe004db8", "folder-name");
        }
    }
}

```

```
        System.out.println(uuid);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

# Mail samples

## Basics

 Example of uuid:

- Using UUID -> "064ff51a-b815-4f48-a096-b4946876784f";


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "login". You can access the "login" method from webservice object "ws" as is shown below:

```
ws.login(user, password);
```

 Once you are logged in with the webservices the session is kept in the webservice Object. Then you can use the other API method

At this point you can use all the Mail methods from "mail" class as is shown below:

```
ws.mail.getProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbcd")
```

## Methods

### getProperties

Description:

Method	Return values	Description
<b>getProperties(String uuid)</b>	<b>Mail</b>	Returns the mail properties.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
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);
            System.out.println(ws.mail.getProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbced"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**delete**

Description:

Method	Return values	Description
<b>delete(String uuid)</b>	<b>void</b>	Deletes a mail.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.delete("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**purge**

Description:

Method	Return values	Description
<b>purge(String uuid)</b>	<b>void</b>	Mail is definitely removed from repository.

Usually, you will purge mails into /okm:trash/userId - the personal trash user locations - but it is possible to directly purge any mail from the whole repository.



When a mail is purged it will only be able to be restored from a previously repository backup. The purge action removes the mail definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.purge("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**rename**

Description:

Method	Return values	Description
<b>rename(String uuid, String newName)</b>	<b>void</b>	Rename a mail.



From OpenKM frontend UI the subject is used to show the mail name at file browser table. That means this change will take effect internally on mail path, but will not be appreciated from default UI.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
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);
            ws.mail.rename("e3831cc4-30f0-419a-8fbf-a3418472ada5", "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**move**

Description:

Method	Return values	Description
<b>move(String uuid, String dstId)</b>	<b>void</b>	Move 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.move("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafbc5d");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**copy**


Description:

Method	Return values	Description
--------	---------------	-------------

<b>public void copy(String uuid, String dstId, String newName)</b>	<b>void</b>	Copies mail into a folder or record.
--	-------------	--------------------------------------

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, mail will preserve the same name.



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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.copy("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafbc5d", "new_
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**extendedCopy**

Description:

Method	Return values	Description
<b>extendedCopy(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security, String newName)</b>	<b>Mail</b>	Copy mail width with associated data into a folder or record.

The values of the dstId parameter should be a folder or record UUID.



By default, only the binary data and the security grants, the metadata, keywords, etc. of the folder are not 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 security parameter is true the original value of the security will be copied.
- When newName is set the mail name is renamed.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Mail mail = ws.mail.extendedCopy("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-");
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getChildren

Description:

Method	Return values	Description
<b>getChildren(String uuid)</b>	<b>List&lt;Mail&gt;</b>	Returns a list of all mails which their parent is fldId.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Mail mail : ws.mail.getChildren("ada67d44-b081-4b23-bdc1-74181cafbc5d")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**isValid**

Description:

Method	Return values	Description
<b>isValid(String uuid)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is a mail or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Return false
            System.out.println(ws.mail.isValid("ada67d44-b081-4b23-bdc1-74181cafbc5d"));

            // Return true
            System.out.println(ws.mail.isValid("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}
}
```

**getPath**

Description:

Method	Return values	Description
<b>getPath(String uuid)</b>	<b>String</b>	Converts mail UUID to mail path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.mail.getPath("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**createAttachment**

Description:

Method	Return values	Description
<b>createAttachment(String uuid, String docName, InputStream is)</b>	<b>Document</b>	Adds an attachment to the mail.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.mail.createAttachment("e3831cc4-30f0-419a-8fbf-a3418472ada5", "logo.png", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**deleteAttachment**

Description:

Method	Return values	Description
<b>deleteAttachment(String uuid, String docId)</b>	<b>void</b>	Delete a mail attachment.
The value of the parameter <b>docId</b> parameter can be a valid document <b>UUID</b> .		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.deleteAttachment("e3831cc4-30f0-419a-8fbf-a3418472ada5", "6dcb02dc-0881-4610-b2bc-a158ed71");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getAttachments**

Description:

Method	Return values	Description
<b>getAttachments(String uuid)</b>	<b>List&lt;Document&gt;</b>	Retrieves a list of all the attachment documents of the mails.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Document doc : ws.mail.getAttachments("e3831cc4-30f0-419a-8fbf-a3418472ada5")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**sendMailWithAttachments**

Description:

Method	Return values	Description
<b>sendMailWithAttachments(List&lt;String&gt; to, List&lt;String&gt; cc, List&lt;String&gt; bcc, List&lt;String&gt; replyTo, String subject, String body, List&lt;String&gt; docsId, String uuid)</b>	<b>void</b>	Send mail message with attachment.

The values of the uuid parameter should be a folder or record UUID. The uuid parameter indicate where the mail will be stored in the repository after is sent.

Other parameters:

- to are a list of mail accounts destination.

- subject is the mail subject.
- cc, bcc, replyTo is a list of mail accounts destination ( optional )
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail ( optional ).

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> to = Arrays.asList("gnu.java.sergio@mail.com");
            List<String> cc = new ArrayList<>();
            List<String> bcc = new ArrayList<>();
            List<String> replyTo = new ArrayList<>();
            List<String> docsId = Arrays.asList("46762f90-82c6-4886-8d21-ad3017dd78a7", "8cd1e072-8595-4dd3-b12
            ws.mail.sendMailWithAttachments(to, cc, bcc, replyTo, "some subject", "some body", docsId, "85f07f05-1641
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### sendMailWithAttachments

Description:

Method	Return values	Description
<b>sendMailWithAttachments(String from, List&lt;String&gt; to, List&lt;String&gt; cc, List&lt;String&gt; bcc, List&lt;String&gt; replyTo, String subject, String body, List&lt;String&gt; docsId, String uuid)</b>	<b>void</b>	Send mail message with attachment.

The values of the uuid parameter should be a folder or record UUID. The uuid parameter indicate where the mail will be stored in the repository after is sent.

Other parameters:

- from is optional and maybe set with empty value
- to are a list of mail accounts destination.
- subject is the mail subject.
- cc, bcc, replyTo is a list of mail accounts destination ( optional )
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail ( optional ).

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String from = "some@nomail.com";
            List<String> to = Arrays.asList("gnu.java.sergio@mail.com");
            List<String> cc = new ArrayList<>();
            List<String> bcc = new ArrayList<>();
            List<String> replyTo = new ArrayList<>();
            List<String> docsId = Arrays.asList("46762f90-82c6-4886-8d21-ad3017dd78a7", "8cd1e072-8595-4dd3-b12
            ws.mail.sendMailWithAttachments(from, to, cc, bcc, replyTo, "some subject", "some body", docsId, "85f07f0f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**importEml**

Description:

Method	Return values	Description
<b>importEml(String uuid, String title, InputStream is)</b>	<b>Mail</b>	Import a mail in EML format.

The values of the **uuid** parameter should be a folder or record UUID. The **uuid** parameter indicates where the mail will be stored in the repository after is imported.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.eml");
            Mail mail = ws.mail.importEml("85f07f05-1641-47e8-891f-0ea30643554e", "some title", is);
            System.out.println(mail);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### importMsg

Description:

Method	Return values	Description
<b>importMsg(String uuid, String title, InputStream is)</b>	<b>Mail</b>	Import a mail in MSG format.

The values of the **uuid** parameter should be a folder or record UUID. The **uuid** parameter indicates where the mail will be stored in the repository after is sent.

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

```

```

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.msg");
            Mail mail = ws.mail.importMsg("85f07f05-1641-47e8-891f-0ea30643554e", "some title", is);
            System.out.println(mail);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### setTitle

Description:

Method	Return values	Description
<b>setTitle(String uuid, String title)</b>	<b>void</b>	Sets the mail title.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.setTitle("9d5dd110-db99-4d72-8cf7-610b027a4822", "new title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**sendMail**

Description:

Method	Return values	Description
<b>sendMail(List&lt;String&gt; recipients, String subject, String body)</b>	<b>void</b>	Sends a mail.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            StringBuffer body = new StringBuffer();
            body.append("Test message body.");
            List<String> recipients = new ArrayList<>();
            recipients.add("some@mail.com");
            ws.mail.sendMail(recipients, "Testing sending mail from OpenKM", body.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**sendMail**

Description:

Method	Return values	Description
<b>sendMail(String from, List&lt;String&gt; recipients, String subject, String body)</b>	<b>void</b>	Sends a mail.
Other parameters:		
<ul style="list-style-type: none"> <li>from is optional and maybe set with empty value</li> </ul>		

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String from = "some@nomail.com";
            StringBuffer body = new StringBuffer();
            body.append("Test message body.");
            List<String> recipients = new ArrayList<>();
            recipients.add("some@mail.com");
            ws.mail.sendMail(to, recipients, "Testing sending mail from OpenKM", body.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## sendMail

Description:

Method	Return values	Description
<b>sendMail(String to, String subject, String body)</b>	<b>void</b>	Send a mail.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Test {

    private static final Logger log = LoggerFactory.getLogger(Test.class);

    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);

```

```

        // sendMail
        String to = "some@nomail.com";
        StringBuffer body = new StringBuffer();
        body.append("message test");
        ws.mail.sendMail(to, "sigma subject", body.toString());
    } catch (Exception e) {
        log.error(e.getMessage());
    }
}
}

```

**setDispositionStage**

Description:

Method	Return values	Description
<b>setDispositionStage(String uuid, long stage)</b>	<b>void</b>	Set the disposition stage

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stage = 1;
            ws.mail.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setDescription**

Description:

Method	Return values	Description
<b>setDescription(String uuid, String description)</b>	<b>void</b>	Set the description.

Example:

---

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.mail.setDescription("b405a504-d8cb-4166-ac51-22f68acee8c5", "some description");
            Mail mail = ws.mail.getProperties("b405a504-d8cb-4166-ac51-22f68acee8c5");
            System.out.println("Description: " + mail.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getContent**

Description:

Method	Return values	Description
<b>getContent(String mailId)</b>	<b>InputStream</b>	Retrieve mail content - binary data - of the current mail version.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            OutputStream fos = new FileOutputStream("/home/openkm/test.eml");
            InputStream is = ws.mail.getContent("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
<b>createMail(String uuid, String title, InputStream is, String type)</b>	<b>WizardNode</b>	Create a new document with a wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

Available types:

- Mail.ORIGIN\_MSG
- Mail.ORIGIN\_EML

**i** The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    InputStream is = new FileInputStream("/home/openkm/sample2.eml");
    WizardNode wn = ws.mail.createWizardl("4c195453-246b-4ce9-86ba-b84e68d1f284", "test title", is, Mail.OTHER);
    System.out.print(wn);
    IOUtils.closeQuietly(is);
} catch (Exception e) {
    e.printStackTrace();
}
}
```


**getThumbnail**

Description:

Method	Return values	Description
<b>getThumbnail(String mailId, ThumbnailType type)</b>	<b>InputStream</b>	Returns thumbnail image data.

Available types:

- ThumbnailType.THUMBNAIL\_PROPERTIES ( shown in properties view )
- ThumbnailType.THUMBNAIL\_LIGHTBOX ( shown in light box )
- ThumbnailType.THUMBNAIL\_SEARCH ( shown in search view )

 Each thumbnail type has its own image dimensions.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ThumbnailType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    OutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/thumbnail.png");
    InputStream is = ws.mail.getMailThumbnail("1778f0b5-672c-48c1-b54e-494c18dd6df4", ThumbnailType.THUMBNAIL);
    IOUtils.copy(is, fos);
    IOUtils.closeQuietly(is);
    IOUtils.closeQuietly(fos);
} catch (Exception e) {
    e.printStackTrace();
}
}
}
    
```


**getMailsPaginated**

Description:

Method	Return values	Description
<b>getMailsPaginated(String context, int offset, int limit, MailFilterQuery filter, String orderColumn, bool orderAsc)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of emails paginated results filtered by the values of the MailFilterQuery 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" says to skip that many results before the beginning to return results.
- The parameter "orderColumn" can have the following values: uuid, subject, from, sentDate and hasAttachments.
- The parameter "orderAsc" can be "true" in case of ascending order and "false" otherwise.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.AttachmentEnum;
import com.openkm.sdk4j.bean.MailFilterQuery;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            MailFilterQuery filter = new MailFilterQuery();
            filter.setSubject("");
            filter.setAttachments(AttachmentEnum.ALL);
            String orderColumn = "subject";
            SimpleNodeBaseResultSet simpleNodeBaseResultSet = ws.mail.getMailsPaginated("/okm:root", 0, 20, filter);

            for (SimpleNodeBase simpleNodeBase : simpleNodeBaseResultSet.getResults()) {
                System.out.println(simpleNodeBase);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## getAccounts

Description:

Method	Return values	Description
<b>getAccounts()</b>	<b>List&lt;MailAccount&gt;</b>	Retrieves a list of user email accounts.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailAccount;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    for (MailAccount mailAccount : ws.mail.getAccounts()) {
        System.out.println(mailAccount);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**getMailMessages**

Description:

Method	Return values	Description
<b>getMailMessages(long accountId, long start)</b>	<b>MailServerMessages</b>	Retrieves a list of messages in the email server for an email account.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ExternalMail;
import com.openkm.sdk4j.bean.MailServerMessages;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long accountId = 1;
            long start = 1;
            MailServerMessages msg = ws.mail.getMailMessages(accountId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                System.out.println(mail);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**addAccount**

Description:

Method	Return values	Description
<b>addAccount(MailAccount mailAccount)</b>	<b>void</b>	Add an email account.

**i** It is a good practice to create an account, verify the mail configuration with the `testMailAccount(MailAccount mail)`.

When you do **not set the user**, the account will be **added by default to the logged user**.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailAccount;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            MailAccount mailAccount = new MailAccount();
            mailAccount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
            mailAccount.setMailUser("test@none.com");
            mailAccount.setMailPassword("123456");
            mailAccount.setMailHost("imap.gmail.com");
            mailAccount.setMailFolder("OpenKM");
            mailAccount.setActive(true);
            mailAccount.setRecursive(true);
            ws.mail.addAccount(mailAccount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**updateAccount**

Description:

Method	Return values	Description
<b>updateAccount(MailAccount mailAccount)</b>	<b>void</b>	Update the main configuration data of an email account.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailAccount;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            MailAccount mailAccount = new MailAccount();
            mailAccount.setId(2);
            mailAccount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
            mailAccount.setMailUser("testupdate@none.com");
            mailAccount.setMailPassword("123456789");
            mailAccount.setMailHost("imap.gmail.com");
            mailAccount.setMailFolder("OpenKM");
            mailAccount.setActive(true);
            mailAccount.setRecursive(true);
            ws.mail.updateAccount(mailAccount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### testAccount

Description:

Method	Return values	Description
<b>testAccount(MailAccount mailAccount)</b>	<b>void</b>	Check email account connection.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailAccount;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
MailAccount mailAccount = new MailAccount();
mailAccount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
mailAccount.setMailUser("testupdate@none.com");
mailAccount.setMailPassword("123456789");
mailAccount.setMailHost("imap.gmail.com");
mailAccount.setMailFolder("OpenKM");
mailAccount.setActive(true);
mailAccount.setRecursive(true);

// Test mail account
ws.mail.testAccount(mailAccount);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**deleteAccount**

Description:

Method	Return values	Description
<b>deleteAccount(long mailAccountId)</b>	<b>void</b>	Delete email account.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long mailAccountId = 2;
            ws.mail.deleteAccount(mailAccountId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**importMessages**

Description:

Method	Return values	Description

<b>importMessages(long mailAccountId, List&lt;Long&gt; messageIds)</b>	<b>void</b>	Import email account messages.
--	-------------	--------------------------------

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ExternalMail;
import com.openkm.sdk4j.bean.MailServerMessages;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<Long> messageIds = new ArrayList<>();

            long mailAccountId = 1; // Valid mailAccountId
            long start = 1;
            MailServerMessages msg = ws.mail.getMailMessages(mailAccountId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                messageIds.add(mail.getUid());
            }

            ws.mail.importMailMessages(mailAccountId, messageIds);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**createFilter**

Description:

Method	Return values	Description
<b>createFilter(long mailAccountId, MailFilter mailFilter)</b>	<b>void</b>	Add email account filter.

Example

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailFilter;
import com.openkm.sdk4j.impl.OKMWebservices;
    
```

```
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);
            long mailAccountId = 2; // Valid mailAccountId

            MailFilter filter = new MailFilter();
            filter.setOrder(0);
            filter.setGrouping(false);
            filter.setExclusive(true);
            filter.setActive(false);
            filter.setNode("a9d5c158-c7b1-4771-b9c3-b8d24f5a2645");

            ws.mail.createFilter(mailAccountId, filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**updateFilter**

Description:

Method	Return values	Description
<b>updateFilter(MailFilter mailFilter)</b>	<b>void</b>	Update email account filter.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailFilter;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            MailFilter filter = new MailFilter();
            filter.setId(2); // Valid filter id
            filter.setOrder(0);
            filter.setGrouping(false);
            filter.setExclusive(true);
            filter.setActive(true);
            filter.setNode("a9d5c158-c7b1-4771-b9c3-b8d24f5a2645");
        }
    }
}
```

```

        ws.mail.updateFilter(filter);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**deleteFilter**

Description:

Method	Return values	Description
<b>deleteFilter(long mailFilterId)</b>	<b>void</b>	Delete email account filter.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);


            long filterId = 1; // Valid filter id
            ws.mail.deleteFilter(filterId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**createRule**

Description:

Method	Return values	Description
<b>createRule(long filterId, MailFilterRule rule)</b>	<b>void</b>	Create email filter rule.



Mail account filter parameters:

- **Field:** Sets the mail field that will be evaluated by the rule.



Available options are:

- MailFilterRule.FIELD\_FROM
- MailFilterRule.FIELD\_TO
- MailFilterRule.FIELD\_SUBJECT
- MailFilterRule.FIELD\_CONTENT
- MailFilterRule.FIELD\_ATTACHMENT

- **Operation:** Set the operation that will be done for the evaluation process.



Available options are:

- MailFilterRule.OPERATION\_EQUALS
- MailFilterRule.OPERATION\_NOT\_EQUALS
- MailFilterRule.OPERATION\_CONTAINS
- MailFilterRule.OPERATION\_ENDS\_WITH
- MailFilterRule.OPERATION\_STARTS\_WITH

- **Value:** Sets the value that will be used for the evaluation process.
- **Active:** Sets rule enabled or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailFilterRule;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            long filterId = 2;// Valid filter id

            MailFilterRule rule = new MailFilterRule();
            rule.setOperation(MailFilterRule.OPERATION_CONTAINS);
            rule.setValue("test");
            rule.setField(MailFilterRule.FIELD_FROM);
```

```

rule.setActive(false);


ws.mail.createRule(filterId, rule);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```


**updateRule**

Description:

Method	Return values	Description
<b>updateMailRule(MailFilterRule rule)</b>	<b>void</b>	Update email coount rule.


 Mail account filter parameters:

- **Field:** Sets the mail field that will be evaluated by the rule.

 Available options are:

- MailFilterRule.FIELD\_FROM
- MailFilterRule.FIELD\_TO
- MailFilterRule.FIELD\_SUBJECT
- MailFilterRule.FIELD\_CONTENT
- MailFilterRule.FIELD\_ATTACHMENT

- **Operation:** Set the operation that will be done for the evaluation process.

 Available options are:

- MailFilterRule.OPERATION\_EQUALS
- MailFilterRule.OPERATION\_NOT\_EQUALS
- MailFilterRule.OPERATION\_CONTAINS
- MailFilterRule.OPERATION\_ENDS\_WITH
- MailFilterRule.OPERATION\_STARTS\_WITH

- **Value:** Sets the value that will be used for the evaluation process.
- **Active:** Sets rule enabled or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailFilterRule;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            MailFilterRule rule = new MailFilterRule();
            rule.setId(2);
            rule.setOperation(MailFilterRule.OPERATION_EQUALS);
            rule.setValue("test");
            rule.setField(MailFilterRule.FIELD_FROM);
            rule.setActive(false);

            ws.mail.updateRule(rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**deleteRule**

Description:

Method	Return values	Description
<b>deleteRule(long ruleId)</b>	<b>void</b>	Delete a rule of an email account.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        }
    }
}
    
```

```

        long ruleId = 1; // Valid rule id
        ws.mail.deleteRule(ruleId);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getFilterRules**

Description:

Method	Return values	Description
<b>getFilterRules(long filterId)</b>	<b>List&lt;MailFilterRule&gt;</b>	Retrieve a list rules by filter.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MailFilterRule;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            long filterId = 2; // Valid filter id
            for (MailFilterRule mailFilterRule : ws.mail.getFilterRules(filterId)) {
                System.out.println(mailFilterRule);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**forwardEmail**

Description:

Method	Return values	Description
<b>forwardEmail(String uuid, List&lt;String&gt; users, List&lt;String&gt; roles, List&lt;String&gt; mails, String message)</b>	<b>void</b>	Forward a email to a list of users, roles or external emails.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            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");

            ws.mail.forwardEmail(mailUuid, users, roles, mails, "Any message");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## getPdf

Description:

Method	Return values	Description
<b>getPdf(String uuid)</b>	<b>InputStream</b>	Returns a PDF of the email.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
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);
            Mail mail = ws.mail.getProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
            InputStream is = ws.mail.getPdf(mail.getUuid());
            OutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/" + mail.getSubject() + ".pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**saveAsPdf**

Description:

Method	Return values	Description
<b>saveAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)</b>	<b>Document</b>	Save the mail as PDF in the OpenKM repository.
<p>The value of the <b>uuid</b> parameter should be a Mail UUID.</p> <p>When the parameter newName value is null, the document will preserve the same name.</p> <div style="border: 1px dashed #ccc; background-color: #e0f0ff; padding: 10px; margin-top: 10px;"> <p><b>i</b> Additional:</p> <ul style="list-style-type: none"> <li>• When the propertyGroups parameter is true the metadata groups of the source are copied to the target.</li> <li>• When the security parameter is true the security of the source is copied to the target.</li> </ul> </div>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        Mail mail = ws.mail.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
        Document docPdf = ws.mail.saveAsPdf(mail.getUuid(), mail.getSubject() + ".pdf", true, true);
        System.out.println("Document pdf: " + docPdf.getPath());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

## Node samples

### Basics



Example of uuid:

- Using UUID -> "373bcdd0-c082-4e7b-addr-e10ef813946e";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "login". You can access the "login" method from webservice object "ws" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Node methods from "node" class as is shown below:

```
ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb")
```

### Methods

#### getNodeByUuid

Description:

Method	Return values	Description
<code>getNodeByUuid(String uuid)</code>	<b>Node</b>	Get a node by uuid.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```

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);
            Node node = ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getVersionHistory

Description:

Method	Return values	Description
<b>getVersionHistory(String uuid)</b>	<b>List&lt;Version&gt;</b>	Returns a list of the version history of a document.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Version;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<Version> versions = ws.node.getVersionHistory("3767deb4-21e7-4272-82be-fece5384fbab");
            for (Version version : versions) {
                System.out.println(version);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### restoreVersion

Description:

Method	Return values	Description
<b>restoreVersion(String uuid, String versionName)</b>	<b>void</b>	Restore a document to a specific version.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.restoreVersion("3767deb4-21e7-4272-82be-fece5384fbab", "1.2");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### deleteVersion

Description:

Method	Return values	Description
<b>deleteVersion(String uuid, String versionName)</b>	<b>void</b>	Delete a specific version.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

```

```

        ws.node.deleteVersion("5aab162d-df4f-4392-96c9-8fc1607e3903", "1.1");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**purgeVersionHistory**

Description:

Method	Return values	Description
<b>purgeVersionHistory(String uuid)</b>	<b>void</b>	Purge version history of a document.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.purgeVersionHistory("3767deb4-21e7-4272-82be-feece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**maybePromotedAsRecord**

Description:

Method	Return values	Description
<b>maybePromotedAsRecord(String uuid, boolean fullEvaluation)</b>	<b>PromoteAsRecordEvaluation</b>	Returns a PromoteAsRecordEvaluation object.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PromoteAsRecordEvaluation;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            PromoteAsRecordEvaluation pre = ws.node.maybePromotedAsRecord("3767deb4-21e7-4272-82be-fece5384fbab");
            System.out.println(pre);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**promoteAsRecord**

Description:

Method	Return values	Description
<b>promoteAsRecord(String uuid)</b>	<b>void</b>	Promote as record.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.promoteAsRecord("3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**degradeRecord**

Description:

Method	Return values	Description
<b>degradeRecord(String uuid)</b>	<b>void</b>	Degrade a record.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.degradeRecord("3767deb4-21e7-4272-82be-feece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**isElectronicRecordPath**

Description:

Method	Return values	Description
<b>isElectronicRecordPath(String uuid)</b>	<b>boolean</b>	Returns true when the node is into an electronic record.


Return true when one of the parents of the node is an electronic record.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    System.out.println(ws.node.isElectronicRecordPath("3767deb4-21e7-4272-82be-fece5384fbab"));
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### getElectronicRecordInPath

Description:

Method	Return values	Description
<b>getElectronicRecordInPath(String uuid)</b>	<b>Record</b>	Get the electronic record in the path.

 Returns the first electronic record in the path of the node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Record record = ws.node.getElectronicRecordInPath("3767deb4-21e7-4272-82be-fece5384fbab");
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getChildrenNodesPaginated

Description:

Method	Return values	Description
<b>getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List&lt;Integer&gt; filteredTypes)</b>	<b>SimpleNodeBaseList</b>	Get children nodes paginated.

**i** Available filteredTypes values:

```
SimpleNodeBase.TYPE_FOLDER
SimpleNodeBase.TYPE_DOCUMENT
SimpleNodeBase.TYPE_MAIL
SimpleNodeBase.TYPE_RECORD
```

You should do:

```
List<Integer> filteredTypes = new ArrayList<>();
filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
```

**✓** 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.

**i** 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
```

```

import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
            filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesPaginated(uuid, 0, 10, "", NodesPaginationInfo.C
                filteredTypes);
            System.out.println("Filtered Elements: " + listNode.getFilteredElements());
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getChildrenNodesPaginated

Description:

Method	Return values	Description
<b>getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List&lt;Integer&gt; filteredTypes, String pluginName)</b>	<b>SimpleNodeBaseList</b>	Get children nodes paginated.

**i** Available filteredTypes values:

SimpleNodeBase.TYPE\_FOLDER  
 SimpleNodeBase.TYPE\_DOCUMENT  
 SimpleNodeBase.TYPE\_MAIL  
 SimpleNodeBase.TYPE\_RECORD

You should do:

```
List<Integer> filteredTypes = new ArrayList<>();
filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
```



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

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
            filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesPaginated(uuid, 0, 10, "", NodesPaginationInfo.C
                filteredTypes, "");
            System.out.println("Filtered Elements: " + listNode.getFilteredElements());
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodeList()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```


    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```


### getChildrenNodesByCategoryPaginated

Description:

Method	Return values	Description
<b>getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List&lt;Integer&gt; filteredTypes)</b>	<b>SimpleNodeBaseList</b>	Get children nodes by category paginated.

 The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

```

```

import java.util.ArrayList;
import java.util.List;

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);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(1);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesByCategoryPaginated(uuid, 0, 10, "", NodesPagei
                filteredTypes);
            System.out.println("Filtered Elements: " + listNode.getFilteredElements());
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**getChildrenNodesByCategoryPaginated**

Description:

Method	Return values	Description
<b>getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List&lt;Integer&gt; filteredTypes, String pluginName)</b>	<b>SimpleNodeBaseList</b>	Get children nodes by category paginated.

 The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(1);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesByCategoryPaginated(uuid, 0, 10, "", NodesPageInfo
                filteredTypes, "");
            System.out.println("Filtered Elements: " + listNode.getFilteredElements());
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getBreadcrumb**

Description:

Method	Return values	Description
<b>getBreadcrumb(String uuid)</b>	<b>List&lt;BreadcrumbItem&gt;</b>	Get breadcrumb.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.BreadCrumbItem;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<BreadCrumbItem> list = ws.node.getBreadcrumb("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
<b>subscribe(String uuid)</b>	<b>void</b>	Adds a subscription to a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.subscribe("39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}

```

## unsubscribe

Description:

Method	Return values	Description
<b>unsubscribe(String uuid)</b>	<b>void</b>	Delete a subscription to a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.unsubscribe("39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## importZip

Description:

Method	Return values	Description
<b>importZip(String uuid, InputStream is)</b>	<b>void</b>	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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
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);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/import.zip");
            ws.node.importZip("212e7c1f-443d-4aac-a12c-0b818ca03419", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**unZip**

Description:

Method	Return values	Description
<b>unZip(String uuid, String dstId)</b>	<b>void</b>	Unzip file.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

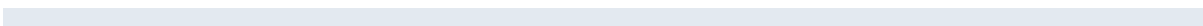
public class Test {

    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);
            String uuid = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801"; // zip File
            String dstId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7"; // destination uuid
            ws.node.unZip(uuid, dstId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**exportZip**

Description:



Method	Return values	Description
<b>exportZip(List&lt;String&gt; uuids, boolean withPath, boolean background)</b>	<b>InputStream</b>	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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> uuids = new ArrayList<>();
            uuids.add("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            uuids.add("d386cff8-1d4d-472f-9c6d-f21955ec499a");

            OutputStream fos = new FileOutputStream("/home/openkm/export.zip");

            InputStream is = ws.node.exportZip(uuids, true, true);
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getNodeFiltered

Description:

Method	Return values	Description
<b>getNodeFiltered(List&lt;String&gt; uuids)</b>	<b>List&lt;Node&gt;</b>	Return a list of nodes.

Example:

```

package com.openkm;

```

```

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> uuids = new ArrayList<>();
            uuids.add("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            uuids.add("d386cff8-1d4d-472f-9c6d-f21955ec499a");

            List<Node> nodes = ws.node.getNodesFiltered(uuids);
            for (Node node : nodes) {
                System.out.println(node);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**evaluateDownloadZip**

Description:

Method	Return values	Description
<b>evaluateDownloadZip(List&lt;String&gt; uuids)</b>	<b>ZipDownloadEvaluationResult</b>	Return a ZipDownloadEvaluationResult object.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ZipDownloadEvaluationResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    List<String> uuids = new ArrayList<>();
    uuids.add("0f6463f3-4d36-4091-b518-4fe7c353ee70");
    uuids.add("d386cff8-1d4d-472f-9c6d-f21955ec499a");

    ZipDownloadEvaluationResult result = ws.node.evaluateDownloadZip(uuids);
    System.out.println(result);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**generateDownloadToken**

Description:

Method	Return values	Description
<b>generateDownloadToken(String uuid, boolean preview)</b>	<b>String</b>	Generate a node download link.

**i** When parameter preview is set to true, the token will be generated for previewing purpose.  
 The preview token expires by default in a minute.  
 The user must build the download URL with the returned token:

- Download: <http://localhost:8080/openkm/download?DTK=token>
- Preview: <http://localhost:8080/openkm/download?PTK=token>

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.node.generateDownloadToken("b2f88679-e3fd-4f97-bf0e-abf76f9ec499", true));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}
}
```

**restore**

Description:

Method	Return values	Description
<b>restore(String uuid)</b>	<b>Node</b>	Restore a node

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Node node = ws.node.restore("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**hasNodesLockedByOtherUser**

Description:

Method	Return values	Description
<b>hasNodesLockedByOtherUser(String uuid)</b>	<b>boolean</b>	If the node is blocked by other users

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        if(ws.node.hasNodesLockedByOtherUser("0f6463f3-4d36-4091-b518-4fe7c353ee70")) {
            System.out.println("Is blocked");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**setComment**

Description:

Method	Return values	Description
<b>setComment(String uuid, String versionName, String comment)</b>	<b>void</b>	Sets the comment for a specific node version.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.node.setComment("1ec49da9-1746-4875-ae32-9281d7303a62", "1.14", "Update comment");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getPaginatorConfig**

Description:

Method	Return values	Description
--------	---------------	-------------

<b>getPaginatorConfig(String pluginName)</b>	<b>NodePaginatorConfig</b>	Return a NodePaginatorConfig object.
--	----------------------------	--------------------------------------

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodePaginatorConfig;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            NodePaginatorConfig nodePaginatorConfig = ws.node.getPaginatorConfig("com.openkm.plugin.paginate.Te
            System.out.println(nodePaginatorConfig);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getPaginatorPlugins

Description:

Method	Return values	Description
<b>getPaginatorPlugins()</b>	<b>PaginatorPluginList</b>	Return a PaginatorPluginList object.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PaginatorPlugin;
import com.openkm.sdk4j.bean.PaginatorPluginList;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    
```

```
PaginatorPluginList list = ws.node.getPaginatorPlugins();
for (PaginatorPlugin paginatorPlugin : list.getPaginatorPlugins()) {
    System.out.println("Name: " + paginatorPlugin.getName());
    System.out.println("ObjClass: " + paginatorPlugin.getObjClass());
}
} catch (Exception e) {
    e.printStackTrace();
}
}
```

## Note samples

### Basics



Example of uuid:

- Using UUID -> "373bccdd0-c082-4e7b-addr-e10ef813946e";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Note methods from "**note**" class as is shown below:

```
ws.note.add("373bccdd0-c082-4e7b-addr-e10ef813946e", "the note text");
```

### Methods

#### add

Description:

Method	Return values	Description
<b>add(String uuid, String text)</b>	<b>Note</b>	Adds a note to a node and returns an object Note.

Example:

```
package com.openkm;
import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
```


```
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);
            ws.note.add("373bccdd0-c082-4e7b-addd-e10ef813946e", "the note text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**get**

Description:

Method	Return values	Description
<b>get(String noteId)</b>	<b>Note</b>	Retrieves the note.

 The noteId is an UUID.

The object Note has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

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);
            List<Note> notes = ws.note.list("373bccdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                System.out.println(ws.get(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
}

```

**delete**

Description:

Method	Return values	Description
<b>delete(String noteId)</b>	<b>Note</b>	Delete a note.

**i** The noteId is an UUID.

The object Note has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

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);
            List<Note> notes = ws.note.list("373bccdd0-c082-4e7b-addr-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.delete(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**set**

Description:

Method	Return values	Description
<b>set(String noteId, String text)</b>	<b>void</b>	Changes the note text.

**i** The noteId is an UUID.  
The object Node has a variable named path, in that case the path contains an UUID.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

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);
            List<Note> notes = ws.note.list("373bccd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.set(notes.get(0).getPath(), "text modified");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**list**

Description:

Method	Return values	Description
<b>list(String uuid)</b>	<b>List&lt;Note&gt;</b>	Retrieves a list of all the notes of a node.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

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);
    List<Note> notes = ws.note.list("373bccdd0-c082-4e7b-addr-e10ef813946e");
    for (Note note : notes) {
        System.out.println(note);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### getNoteHistories

Description:

Method	Return values	Description
<b>getNoteHistories(String uuid)</b>	<b>List&lt;NoteHistory&gt;</b>	Return the historical notes of a node.

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NoteHistory;

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);
            List<NoteHistory> list = ws.note.getNoteHistories("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573");
            for (NoteHistory noteHistory : list) {
                System.out.println(noteHistory);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

# Notification samples

## Basics


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Notification methods from "**notification**" class as is shown below:


```
ws.notification.notify(uuids, users, roles, mails, message, false);
```

## Methods

### notify

Description:

Method	Return values	Description
<b>notify(List&lt;String&gt; uuids, List&lt;String&gt; users, List&lt;String&gt; roles, List&lt;String&gt; mails, String message, boolean attachment)</b>	<b>void</b>	Send a mail notification.

 The parameter uuids are the UUID of the node ( document, folder, mail or record ).

The parameter users are a set of OpenKM users to be notified.

The parameter roles are a set of OpenKM roles to be notified.

The parameter mails are a set of email addresses - usually external mails - to be notified.

The parameter message is the content body of the mail.

When attachment value is true the node is attached into the mail.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<String> uuids = new ArrayList<>();
            uuids.add("b153c4b7-3d1c-4589-bd42-0ed0f34fd338");

            List<String> users = new ArrayList<>();
            users.add("test");
            users.add("sochoa");

            List<String> roles = new ArrayList<>();
            roles.add("ROLE_TEST");

            List<String> mails = new ArrayList<>();
            String message = "Body of the message";
            ws.notification.notify(uuids, users, roles, mails, message, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## PDF samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is kept in the webservice Object. Then you can use the other API method

At this point you can use all the Pdf methods from "**pdf**" class as is shown below:

```
InputStream is = ws.pdf.getImage("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", 1);
```

### Methods

#### getImage

Description:

Method	Return values	Description
<b>getImage(String uuid, int page, String size)</b>	<b>InputStream</b>	Returns the image of a page.



The document must be a PDF or convertible to PDF.

The default value of size parameter is "150". The value is in pixels.

The value of the **uuid** is the UUID of the document.

The **page** is the number of the page in the document starting from the value "1".

The **size** value is optional. When set is used to resize the image of the page, otherwise is used the default value.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);


            InputStream is = ws.pdf.getImage("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", 1, null);
            OutputStream fos = new FileOutputStream("/home/openkm/page-1.png");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**split**

Description:

Method	Return values	Description
<b>split(String uuid, String dstId, String baseName, List&lt;Integer&gt; pages)</b>	<b>boolean</b>	Return true when the pages of the document have been split.


The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of **baseName** is the base name to create a split document, for example, if baseName value is "test" and pages value is "1,3", the files created will be "test-001.pdf" and "test-003.pdf".

The value of **pages** is the number of pages to be split.

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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {


    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            boolean value = ws.pdf.split("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c10-e7f27c");
            if (value) {
                System.out.println("split done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**extract**

Description:

Method	Return values	Description
<b>extract(String uuid, String dstId, String name, List&lt;Integer&gt; pages)</b>	<b>boolean</b>	Return true when the chosen pages of the document have been extracted.


The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination ( folder or record ).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be extracted.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {


    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            //extract
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c10-e7f27c731442", "e
                System.out.println("extract done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**remove**

Description:

Method	Return values	Description
<b>remove(String uuid, String dstId, String name, List&lt;Integer&gt; pages)</b>	<b>boolean</b>	Return true when the pages of the document have been removed.


The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination ( folder or record ).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be removed.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            // remove
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c10-e7f27c731442", "re
                System.out.println("remove done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**rotate**

Description:

Method	Return values	Description
<b>rotate(String uuid, String dstId, String name, String angle, List&lt;Integer&gt; pages)</b>	<b>boolean</b>	Return true when the pages of the document have been rotated.


The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination ( folder or record ).

The value of **name** is the name of the new document with extracted pages.

The value of **angle** is rotation applied in the chosen pages..

The value of **pages** is the number of pages to be rotated.

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {


    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            // rotate
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.rotate("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c10-e7f27c731442", "rot
                pages)) {
                System.out.println("rotation done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**insertPages**

Description:

Method	Return values	Description
<b>insertPages(String uuid, String srcId, int sourceFromPage, int sourceToPage, int insertFromPage)</b>	<b>boolean</b>	Returns true when the pages have been inserted into the document.


The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the target document where the pages will be inserted.

The value of **srcId** is the UUID of the source document from which the pages will be extracted.

The value of **sourceFromPage** is the starting page number in the source document for page extraction.

The value of **sourceToPage** is the ending page number in the source document for page extraction. Extraction is inclusive of this page.

The value of **insertFromPage** is the page number in the target document where the extracted pages will be inserted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {


    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            ws.pdf.insertPages("6a54d98b-6454-4484-8172-8359a2b4f2f2", "a573da75-ba1e-4f5b-98d2-fbb7da5f1cf1",
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**optimize**

Description:

Method	Return values	Description
<b>optimize(String uuid)</b>	<b>boolean</b>	returns true when optimized successfully

 The uuid of the node must be a PDF document.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            boolean success = ws.pdf.optimize("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
            System.out.println("optimize success: " + success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Plugin samples

### Basics


#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Plugin methods from "**plugin**" class as is shown below:


```
ws.plugin.executePluginPost("com.openkm.plugin.rest.TestRestPlugin", parameters, String.class, null);
```

### Methods

#### executePluginPost

Description:

Method	Return values	Description
<b>executePluginPost(String className, Map&lt;String, String&gt; parameters, Class&lt;?&gt; clazz, InputStream is)</b>	<b>Object</b>	Return the Object value of execution of a class which implements RestPlugin.



- The "className" value must be the canonical class name of a class which implements RestPlugin interface.
- The "parameters" map are values what will be used by class set in className.
- The "clazz" value should be the Class class of the returned object. It's used by REST for unmarshalling the result of the query.
- The "is" stream allow to uploading a document.

 This method execute a REST call at POST.

Example:

```

package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> parameters = new HashMap<>();
            parameters.put("param1", "value1");
            parameters.put("param2", "value2");
            String value = (String) ws.plugin.executePluginPost("com.openkm.plugin.rest.TestRestPlugin", parameters, {
            System.out.println(value);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
    
```

**executePostPluginReturnFile**

Description:

Method	Return values	Description
<b>executePluginPostReturnFile(String className, Map&lt;String, String&gt; parameters, InputStream is)</b>	<b>InputStream</b>	Return an InputStream of execution of a class which implements RestPlugin.

-  • The "className" value must be the canonical class name of a class which implements RestPlugin interface.
- The "parameters" map are values what will be used by class set in className.
- The "is" stream allow to uploading a document.



This method execute a REST call at POST.

The RestPlugin must return a Response object. Take a look at the next sample implementation of the RestPlugin:

```

package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestPlugin.class);

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(isContent);
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
    }
}

```

```

    }
}

```

Example:

```

package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            InputStream is = ws.plugin.executePluginPostReturnFile("com.openkm.plugin.rest.TestGetDocumentRestPlu");
            OutputStream fos = new FileOutputStream("/home/user/Desktop/test.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**executePluginGet**

Description:

Method
<b>executePluginGet(String className, Map&lt;String, String&gt; parameters, Class&lt;?&gt; clazz)</b>

- 
  - The "className" value must be the canonical class name of a class which implements RestPlugin interface
  - The "parameters" map are values what will be used by class set in className.
  - The "clazz" value should be the Class class of the returned object. It's used by REST for unmarshalling the



This method execute a REST call at GET.

You can also execute directly from the browser, take the next url as a sample:

```
http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?className=com.openkm.plugin.
```

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Browser URL
            // http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?className=com.openkm.plugin.rest
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            parameters.put("docId2", "/okm:root/invoices/00000001_001_of_001.pdf2");
            String value = (String) ws.plugin.executePluginGet("com.openkm.plugin.rest.TestRestPlugin", parameters, S
            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**executePluginAtGetReturnFile**

Description:

Method
<b>executeGetPluginReturnFile(String className, Map&lt;String, String&gt; parameters)</b>



- The "className" value must be the canonical class name of a class which implements RestPlugin interface
- The "parameters" map are values what will be used by class set in className.



This method execute a REST call at GET.

The RestPlugin must return a Response object. Take a look at the next sample implementation of the RestPlugin:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestPlugin.class);

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(isContent);
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
    }
}
```

```
    }
}
```

You can also execute directly from the browser, take the next url as a sample:

```
http://localhost:8080/OpenKM/services/rest/plugin/executeGetPluginReturnFile?className=com.open
```

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Browser URL
            // http://localhost:8180/OpenKM/services/rest/plugin/executeGetPluginReturnFile?className=com.openkm.i
            // http://localhost:8180/OpenKM/services/rest/plugin/executeGetPluginReturnFile?className=com.openkm.i
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            InputStream is = ws.plugin.executePluginGetReturnFile("com.openkm.plugin.rest.TestGetDocumentRestPlu
            OutputStream fos = new FileOutputStream("/home/jllort/Escritorio/test.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Property samples

### Basics

The value of this parameter can be a valid document, folder, mail or record **UUID**.



Example of nodeId:

- Using UUID -> "f123a950-0329-4d62-8328-0ff500fd42db";


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Repository methods from "**repository**" class as is shown below:

```
ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-bd78-e26d7c6fb648");
```

### Methods

#### addCategory

Description:

Method	Return values	Description
<b>addCategory(String uuid, String catId)</b>	<b>void</b>	Set a relation between a category and a node.
The value of the catId parameter should be a category folder UUID.		

Example:

```
 
```

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-bd78-e26d7c6f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**removeCategory**

Description:

Method	Return values	Description
<b>removeCategory(String uuid, String catId)</b>	<b>void</b>	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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.property.removeCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-bd78-e26d7c6f");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


**addKeyword**

Description:

Method	Return values	Description
<b>addKeyword(String uuid, String keyword)</b>	<b>void</b>	Add a keyword in a node.

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two\_words" ( the character "\_" is used for the junction ).

 We also we suggest you to add keyword in lowercase format, because OpenKM is case sensitive.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.property.addKeyword("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**removeKeyword**

Description:

Method	Return values	Description
<b>removeKeyword(String uuid, String keyword)</b>	<b>void</b>	Removes a keyword from a node.

Example:

```

package com.openkm;
    
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;


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);
            ws.property.removeKeyword("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**setEncryption**

Description:

Method	Return values	Description
<b>setEncryption(String uuid, String cipherName)</b>	<b>void</b>	Marks a document as an encrypted binary data in the repository.
<p>The parameter nodeId should be a document node.</p> <p>The parameter cipherName saves information about the encryption mechanism.</p> <div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  This method does not perform any kind of encryption, simply mark in the database that a document is encrypted.                 </div>		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        ws.property.setEncryption("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "phrase");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}


```

**unsetEncryption**

Description:

Method	Return values	Description
<b>unsetEncryption(String uuid)</b>	<b>void</b>	Marks a document is a normal binary data into repository.

The parameter **uuid** should be a document node.



This method does not perform any kind of encryption, simply mark into the database that a document has been unencrypted.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.property.unsetEncryption("eee9d70f-6af9-4783-82a7-b8ac94c234b6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setSigned**

Description:

Method	Return values	Description

<b>setSigned(String uuid, boolean signed)</b>	<b>void</b>	Marks a document as signed or unsigned binary data into the repository.
<p>The parameter <b>uuid</b> should be a document node.</p> <div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  This method does not perform any kind of digital signature process, simply mark into the database that a document is signed.                 </div>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.property.setSigned("eee9d70f-6af9-4783-82a7-b8ac94c234b6", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**isSigned**

Description:

Method	Return values	Description
<b>isSigned(String uuid)</b>	<b>boolean</b>	Returns a boolean that indicates if the document is signed or not.
<p>The parameter <b>uuid</b> should be a document node.</p>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
    
```

```
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);  
  
            System.out.println("Is the document signed: " + ws.property.isSigned("6330d2a0-529f-4c14-baa1-ce6a92004  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

## PropertyGroup samples

### Basics



From the older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".

Although we understand this name does not help a lot to identify these methods with metadata ones, for historical reason, we continue maintaining the nomenclature.

For more information about [Metadata](#).



The class `com.openkm.sdk4j.util.ISO8601` should be used to set and parse metadata date fields. The metadata field of type date values is stored in the application in ISO-8601 basic format.

To convert the retrieved metadata field of type date to a valid date use:

```
Calendar cal = ISO8601.parseBasic(metadataFieldValue);
```

To save the date value in the metadata field of type date use:

```
Calendar cal = Calendar.getInstance(); // Present date  
String metadataFieldValue = ISO8601.formatBasic(cal);  
// metadataFieldValue can be saved into repository metadata field of type date
```

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API methods.

At this point you can use all the Property Group methods from "**propertyGroup**" class as is shown below:

```
ws.propertyGroup.addGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", propertiesMap);
```

### Methods

## addGroup

Description:

Method	Return values	Description
<b>addGroup(String uuid, String grpName, Map&lt;String, String&gt; propertiesMap)</b>	<b>void</b>	Adds an empty metadata group to a node.

The grpName should be a valid Metadata group name.



It is not mandatory to set in the propertiesMap parameter all fields values, is enough with the fields you wish to change its values.



The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
"http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
<property-group label="Technology" name="okg:technology">
<select label="Type" name="okp:technology.type" type="multiple">
<option label="Alfa" value="t1"/>
<option label="Beta" value="t2" />
<option label="Omega" value="t3" />
</select>
<select label="Language" name="okp:technology.language" type="simple">
<option label="Java" value="java"/>
<option label="Python" value="python"/>
<option label="PHP" value="php" />
</select>
<input label="Date" name="okp:technology.date" type="date" />
<input label="Comment" name="okp:technology.comment"/>
<textarea label="Description" name="okp:technology.description"/>
<checkbox label="Important" name="okp:technology.important"/>
<input label="Link" type="link" name="okp:technology.link"/>
</property-group>
</property-groups>
```



To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
<option label="Alfa" value="t1"/>
<option label="Beto" value="t2"/>
<option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

Where `"t1"` and `"t2"` are valid values and character `" , "` is used as separator.

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

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);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
            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.language", "[ \"java\" ]");
            properties.put("okp:technology.important", String.valueOf(true));
            ws.propertyGroup.addGroup("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## removeGroup

Description:

Method	Return values	Description
<b>removeGroup(String uuid, String grpName)</b>	<b>void</b>	Removes a metadata group of a node.
The grpName should be a valid Metadata group name.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.propertyGroup.removeGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## getGroups

Description:

Method	Return values	Description
<b>getGroups(String uuid)</b>	<b>List&lt;PropertyGroup&gt;</b>	Retrieves a list of metadata groups assigned to a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        for (PropertyGroup pGroup : ws.propertyGroup.getGroups("8cd1e072-8595-4dd3-b121-41d622c43f08")) {
            System.out.println(pGroup);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getAllGroups**

Description:

Method	Return values	Description
<b>getAllGroups()</b>	<b>List&lt;PropertyGroup&gt;</b>	Retrieves a list of all metadata groups set into the application.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (PropertyGroup pGroup : ws.propertyGroup.getAllGroups()) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

**getPropertyGroupForm**

Description:

Method	Return values	Description
<b>getPropertyGroupForm(String grpName)</b>	<b>List&lt;FormElement&gt;</b>	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

 The method is usually used to display empty form elements for creating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupForm("okg:consulting")) {
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getPropertyGroupFormDefinition**

Description:

Method	Return values	Description
<b>getPropertyGroupFormDefinition(String grpName)</b>	<b>List&lt;FormElement&gt;</b>	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

 The method is usually used to display empty form elements for creating new metadata values.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupFormDefinition("okg:consulting")) {
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getPropertyGroupFormDefinition**

Description:

Method	Return values	Description
<b>getPropertyGroupFormDefinition(String grpName, String uuid)</b>	<b>List&lt;FormElement&gt;</b>	Retrieves a list of all metadata group elements definition.
<p>The grpName should be a valid Metadata group name.</p> <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;"> <p> The method is usually used to display empty form elements for creating new metadata values.</p> </div>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    }
}


```




Method	Return values	Description
<b>setProperties(String uuid, String grpName, Map&lt;String, String&gt; properties)</b>	<b>void</b>	Changes the metadata group values of a node.


The grpName should be a valid Metadata group name.

 Before changing metadata, you **should have the group added in the node** ( see addGroup method ) otherwise will be raised an error.

 It is not mandatory to set in the properties parameterMap all fields values, is enough with the fields you wish to change its values.

 The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
"http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okp:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Date" name="okp:technology.date" type="date" />
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <checkbox label="Important" name="okp:technology.important"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

 To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
  <option label="Alfa" value="t1"/>
  <option label="Beto" value="t2"/>
  <option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

Where `"one"` and `"two"` 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 java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

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);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
            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.language", "[\"java\"]");
            properties.put("okp:technology.important", String.valueOf(true));
            ws.propertyGroup.setProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## hasGroup

Description:

Method	Return values	Description
--------	---------------	-------------

<b>hasGroup(String uuid, String grpName)</b>	<b>Boolean</b>	Returns a boolean that indicate if the node has or not a metadata group.
--	----------------	--

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("Have metadata group:" + ws.propertyGroup.hasGroup("8cd1e072-8595-4dd3-b121-41d6"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getRegisteredDefinition**

Description:

Method	Return values	Description
<b>getRegisteredDefinition()</b>	<b>String</b>	Returns the XML Metada groups definition.


The method only can be executed by super user granted ( ROLE\_ADMIN member user).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        System.out.println(ws.propertyGroup.getRegisteredDefinition());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**registerDefinition**

Description:

Method	Return values	Description
<b>registerDefinition(InputStream is, String name)</b>	<b>void</b>	Set the XML Metada groups definition into the repository.



The method only can be executed by super user granted ( ROLE\_ADMIN member user).

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/PropertyGroups.xml");
            ws.propertyGroup.registerDefinition(is, "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```


**getSuggestions**

Description:

Method	Return values	Description
<b>getSuggestions(String uuid, String grpName, String propName)</b>	<b>List&lt;String&gt;</b>	Retrieves a list of a suggested metadata field values.

 The propName parameter should be a [Metadata Select field](#) type.

 More information at [Creating your own Suggestion Analyzer](#) and [Metadata Select field](#).

 The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
"http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    for (String value : ws.propertyGroup.getSuggestions("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:techno
        System.out.println(value);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**getProperties**

Description:

Method	Return values	Description
<b>getProperties(String uuid, String grpName)</b>	<b>Map&lt;String, String&gt;</b>	Retrieves a map - (key,value) pairs - with Metada group values of a node.

Example:

```

package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> properties = new HashMap<>();
            properties = ws.propertyGroup.getProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:technology")

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getGroupsByVersion**

Description:

Method	Return values	Description
--------	---------------	-------------

<p><b>getGroupsByVersion(String uuid, String versionName)</b></p>	<p><b>List&lt;PropertyGroup&gt;</b></p>	<p>Retrieves a list of metadata groups assigned to a specific version of a node.</p>
---	---	--

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (PropertyGroup pGroup : ws.propertyGroup.getGroupsByVersion("118cb06b-9df7-4d59-bed9-ba986a4f8
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getPropertiesByVersion**

Description:

Method	Return values	Description
<p><b>getPropertiesByVersion(String uuid, String grpName, String versionName)</b></p>	<p><b>Map&lt;String, String&gt;</b></p>	<p>Retrieves a map - (key,value) pairs - with Metadata group values assigned to a specific version of a node.</p>

Example:

```

package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    Map<String, String> properties = new HashMap<>();
    properties = ws.propertyGroup.getPropertiesByVersion("118cb06b-9df7-4d59-bed9-ba986a4f8e0b", "okg:tec


    for (String key : properties.keySet()) {
        System.out.println(key + " > " + properties.get(key));
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**getGroupByVersionForm**

Description:

Method	Return values	Description
<b>getGroupByVersionForm(String uuid, String grpName, String versionName)</b>	<b>List&lt;FormElement&gt;</b>	Retrieves a list of all metadata group elements definition for a specific version of a node.

The grpName should be a valid Metadata group name.

 The method is usually used to display empty form elements for updating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (FormElement fElement : ws.propertyGroup.getGroupByVersionForm("118cb06b-9df7-4d59-bed9-ba986a4f8e0b", "okg:tec")) {
                System.out.println(fElement);
            }
        }
    }
}
```

```

    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getGroup**

Description:

Method	Return values	Description
<b>getGroup(String grpName)</b>	<b>PropertyGroup</b>	Retrieve the metadata group

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.propertyGroup.getGroup("okg:consulting"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getSuggestBoxKeyValue**

Description:

Method	Return values	Description
<b>getSuggestBoxKeyValue(String grpName, String propertyName, String key)</b>	<b>String</b>	Returns the suggestBox value for key.

 The propertyName parameter should be a [Metadata Suggestbox field](#) type.



More information at [Metadata Suggestbox 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 3.7//EN"
"http://www.openkm.com/dtd/property-groups-3.7.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>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
      System.out.println(ws.propertyGroup.getPropertyGroup("okg:consulting"));
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

**getSuggestBoxKeyValuesFiltered**

Description:

Method	Return values	Description
<b>getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)</b>	<b>Map&lt;String, String&gt;</b>	Retrieves a map - (key, value) pairs - with suggestBox values

The propertyName parameter should be a [Metadata Suggestbox field](#) type.



More information at [Metadata Suggestbox 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 3.7//EN"
"http://www.openkm.com/dtd/property-groups-3.7.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>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Map;

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);
            Map<String, String> values = ws.propertyGroup.getSuggestBoxKeyValuesFiltered(Config.GROUP_CONSUI
            for (String key : values.keySet()) {
                String value = values.get(key);
                System.out.println(key + ":" + value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**validateField**

Method	Return values
<code>validateField(String value, String className, List&lt;String&gt; uuids)</code>	String Ret



- The "className" value must be the canonical class name of a class which implements FieldValidator interface



#### Example of the Form validator implementation

More information at [Creating your own Form Validator plugin](#).

In this example it will not be allowed two equal comments in the metadata field named okp:tecnology.comment

```
package com.openkm.plugin.form.validator;

import java.util.ArrayList;
import java.util.List;

import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.form.FieldValidator;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.db.service.LegacySrv;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements FieldValidator {

    @Autowired
    private LegacySrv legacySrv;

    @Autowired
    private OKMRelation okmRelation;

    @Override
    public String getName() {
        return "Duplicated document number";
    }

    @Override
    public String validate(String value, List<String> uuids) {
        String validate = "";

        String token = DbSessionManager.getInstance().getSystemToken();
        String sql = "SELECT RGT_UUID FROM OKM_PGRP_CUR_TECHNOLOGY WHERE RGT_PRO";
        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;
}
```

### Example

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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);
            List<String> uuids = new ArrayList<>();
            uuids.add("26ece92f-9708-4d3f-8033-18dc98b9e7f5");
            uuids.add("7984b622-7c5f-425a-9451-7611c0caf227");

            String value = "test";
            String className = "com.openkm.plugin.form.validator.DuplicateDocumentNumberValidator";
            String message = ws.propertyGroup.validateField(value, className, uuids);
            System.out.println("Validate: " + message);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

# Record samples

## Basics

 Example of uuid:

- Using UUID -> "a66664a3-0e1d-4b03-9049-a2f4732a0802";


### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Record methods from "**record**" class as is shown below:

```
ws.record.create("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new title", 0)
```

## Methods

### create

Description:

Method	Return values	Description
<b>create(String uuid, String name, String title, long nodeClass)</b>	<b>Record</b>	Creates a new record and return as a result an object Record.
The parameters <b>uuid</b> should be any valid folder or record <b>UUID</b> .		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Record record = ws.record.create("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new title", 0);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProperties**

Description:

Method	Return values	Description
<b>getProperties(String uuid)</b>	<b>Record</b>	Returns the record properties.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.record.getProperties("fbe2933e-b141-4557-ab7a-736820ecdb2e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**delete**

Description:

Method	Return values	Description
<b>delete(String uuid)</b>	<b>void</b>	Deletes a record.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.delete("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**purge**

Description:

Method	Return values	Description
<b>purge(String uuid)</b>	<b>void</b>	The record is definitively removed from repository.

Usually you will purge records to /okm:trash/userId - the personal trash user locations - but is possible to directly purge any record from the whole repository.



When a record is purged it will only be able to be restored from a previously repository backup. The purge action remove the record definitively from the repository.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
    
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.purge("fbc2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**rename**

Description:

Method	Return values	Description
<b>rename(String uuid, String newName)</b>	<b>Record</b>	Renames a record.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.rename("5489fc37-3eb7-43de-998c-319725ae0ca0", "new_name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**move**

Description:

Method	Return values	Description

<b>move(String uuid, String dstId)</b>	<b>void</b>	Moves a record to a folder or record.
The values of the dstId parameter should be a folder or record UUID.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.move("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d6950c63");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```


**copy**

Description:

Method	Return values	Description
<b>copy(String uuid, String dstId, String newName)</b>	<b>void</b>	Copies a record to a folder or record.

The values of the dstId parameter should be a folder or record UUID.

When the parameter newName value is null, the record will preserve the same name.



Only the security grants are copied to the destination, the metadata, keywords, etc. of the record are not copied.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        ws.record.copy("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d6950c63", "")
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**isValid**

Description:

Method	Return values	Description
<b>isValid(String uuid)</b>	<b>Boolean</b>	Returns a boolean that indicate if the node is a record or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println("Is a record:" + ws.record.isValid("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

**getChildren**

Description:

Method	Return values	Description
<b>getChildren(String uuid)</b>	<b>List&lt;Record&gt;</b>	Returns a list of all records which their parent is fldId

The parameter uuid can be a folder or a record node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Record rec : ws.record.getChildren("8599eab7-ae61-4628-8010-1103d6950c63")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**setTitle**

Description:

Method	Return values	Description
<b>setTitle(String uuid, String title)</b>	<b>void</b>	Sets a record title.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.setTitle("5489fc37-3eb7-43de-998c-319725ae0ca0", "some title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

```

    }
  }
}

```

**getPath**

Description:

Method	Return values	Description
<b>getPath(String uuid)</b>	<b>String</b>	Converts a record UUID to a record path.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.record.getPath("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setNodeClass**

Description:

Method	Return values	Description
<b>setNodeClass(String uuid, long ncId)</b>	<b>void</b>	Set the NodeClass.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        long nclId = 2;
        ws.record.setNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", nclId);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

### setDispositionStage

Description:

Method	Return values	Description
<b>setDispositionStage(String uuid, long stage)</b>	<b>void</b>	Set the disposition stage

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stage = 1;
            ws.record.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### setDescription

Description:

Method	Return values	Description
<b>setDescription(String uuid, String description)</b>	<b>void</b>	Set a description.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.setDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**extendedCopy**

Description:

Method	Return values	Description
<b>extendedCopy(String uuid, String dstId, String newName, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)</b>	<b>Record</b>	Copies a record 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, the record will preservate the same name.

**i** By default only the binary data and the security grants, the metadata, keywords, etc. of the folder 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 security parameter is true the original value of the security will be copied.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Record record = ws.record.extendedCopy("ada67d44-b081-4b23-bdc1-74181cafb5d", "8599eab7-ae61-46;
                "new name record", true, true, true, true);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**createWizard**

Description:

Method	Return values	Description
<b>createWizard(String uuid, String name, String title, long nodeClass)</b>	<b>WizardNode</b>	Create a new record with wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

**i** The WizardNode contains a list of pending actions what should be done to complete the process of record creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            WizardNode wn = ws.record.createWizard("1f323e88-64ee-4f57-91e2-9276f8c603f9", "PKI-100200", "new,ti
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**createFromTemplate**

Description:

Method	Return values	Description
<b>createFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map&lt;String, String&gt; properties)</b>	<b>Record</b>	Creates a new record from the template and returns an object Record.

The **uuid** parameter is the UUID value of the template file.

The **dstPath** value is the record destination path.

When the template uses metadata groups to fill in fields, then these values are mandatory and must be set into properties parameter.

**i** Additional:

- When category parameter is true the original values of the categories will be copied.
- When keywords parameter is true the original values of the keywords will be copied.
- When notes parameter is true the original values of the notes will be copied.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

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);

            Map<String, String> properties = new HashMap<>();
            // okg:tpl
            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));
            properties.put("okp:tpl.language", "[ \"java\" ]");

            // Property okg:technology
            properties.put("okp:technology.comment", "sdk name");
            Record record = ws.record.createFromTemplate("9a114b17-7e51-41e7-9d3a-dc7b77e37656", "/okm:root/sd
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**createMissingRecords**

Description:

Method	Return values	Description
<b>createMissingRecords(String recPath)</b>	<b>String</b>	Create missing records.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;

```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.record.createMissingRecords("/okm:root/missingrec1/missingrec2/missingrec3");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Relation samples

### Basics



Example of uuid:

- Using UUID -> "f123a950-0329-4d62-8328-0ff500fd42db";

### Suggested code sample

First, you must create the web service object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should log in using the method "login." You can access the "login" method from the web service object "ws" as is shown below:

```
ws.login(user, password);
```



Once logged in with the web services, the session is kept in the web service Object. Then you can use the other API method.

At this point, you can use all the Relation methods from the "relation" class as shown below:

```
ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL);
```

### Methods

#### getRelationTypes

Description:

Method	Return values	Description
<code>getRelationTypes(String type)</code>	<code>List&lt;RelationType&gt;</code>	Retrieves a list of all relations defined of a type.
Available types values: <ul style="list-style-type: none"> <li>• <code>RelationType.BIDIRECTIONAL</code></li> <li>• <code>RelationType.PARENT_CHILD</code></li> <li>• <code>RelationType.MANY_TO_MANY</code></li> </ul>		

 [More information on Relation types.](#)

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationType type : ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL)) {
                System.out.println(type);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**add**

Description:

Method	Return values	Description
<b>add(String nodeAId, String nodeBId, long relTypeId)</b>	<b>void</b>	Sets a relation between two nodes.

The parameters **nodeAId** and **nodeBId** should be any valid document, folder, mail, or record **UUID**.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    for (RelationType type : ws.getRelationTypes(RelationType.BIDIRECTIONAL)) {
        // looking for a relation named invoice
        if (type.getTitle().equals("invoice")) {
            // Relation invoice with budget
            ws.relation.add("46762f90-82c6-4886-8d21-ad3017dd78a7", "8cd1e072-8595-4dd3-b121-41d622c43f0");
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**delete**

Description:

Method	Return values	Description
<b>delete(long relationId)</b>	<b>void</b>	Deletes a relationship.

 Only when a node will not use the relationship it can be deleted. Otherwise, you'll get an error.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Relation;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Relation relation : ws.getRelations("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                // looking for a relation named invoice
                if (relation.getRelationTitle().equals("invoice")) {
                    ws.relation.delete(relation.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
}
}
```

### getRelations

Description:

Method	Return values	Description
<b>getRelations(String uuid)</b>	<b>List&lt;Relation&gt;</b>	Retrieves a list of all relations of a node.
The parameter <b>uuid</b> should be any valid document, folder, mail, or record <b>UUID</b> .		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Relation;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Relation relation : ws.relation.getRelations("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                System.out.println(relation);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### getRelationGroups

Description:

Method	Return values	Description
<b>getRelationGroups(String uuid)</b>	<b>List&lt;RelationGroup&gt;</b>	Retrieves a list of all related groups of a node.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationGroup rg : ws.relation.getRelationGroups("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                System.out.println(rg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**addGroup**

Description:

Method	Return values	Description
<b>addGroup(String uuid, String groupName, long type)</b>	<b>void</b>	Adds a relation group at a node.

A relation group only has the sense to apply a relation type of RelationType.MANY\_TO\_MANY.


More information on [Relation types](#).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationType type : ws.relation.getRelationTypes(RelationType.MANY_TO_MANY)) {
                if (type.getTitle().equals("staple")) {

```

```

        ws.addGroup("46762f90-82c6-4886-8d21-ad3017dd78a7", "staple group", type.getId());
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}


```

**addNodeToGroup**

Description:

Method	Return values	Description
<b>addNodeToGroup(String uuid, long groupId)</b>	<b>void</b>	Adds a node to an existing relation group.

On a relation group, it only has the sense to apply the type RelationType.MANY\_TO\_MANY.

 [More information on Relation types.](#)

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationGroup rg : ws.relation.getRelationGroups("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                if (rg.getName().equals("staple group")) {
                    ws.relation.addNodeToGroup("8f101a85-88e7-4abe-8175-c5fea3e17d8b", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**deleteGroup**

Description:

Method	Return values	Description
<b>deleteGroup(long groupId)</b>	<b>void</b>	Remove the relation group from a node

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationGroup rg : ws.relation.getGroups("8f101a85-88e7-4abe-8175-c5fea3e17d8b")) {
                if (rg.getName().equals("staple group")) {
                    ws.relation.deleteRelationGroup(rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getGroup

Description:

Method	Return values	Description
<b>getGroup(long groupId)</b>	<b>RelationGroup</b>	Finds a relation group by id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    long groupId = 1;
    System.out.println(ws.relation.getGroup(groupId));
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**setGroupName**

Description:

Method	Return values	Description
<b>setGroupName(long groupId, String groupName)</b>	<b>void</b>	Changes the relation group name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long groupId = 1;
            ws.relation.setGroupName(groupId, "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getAllRelationGroups**

Description:

Method	Return values	Description
<b>getAllRelationGroups(int relationTypeId, String filter, int offset, int limit)</b>	<b>RelationGroupResultSet</b>	Retrieves a list of all related groups.



The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" limits 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.bean.RelationGroupResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            int relationTypeId = 2;
            String filter = "";
            RelationGroupResultSet resultSet = ws.relation.getAllRelationGroups(relationTypeId, filter, 0, 10);
            System.out.println("Total: " + resultSet.getTotal());
            for (RelationGroup relationGroup : resultSet.getResults()) {
                System.out.println(relationGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### **deleteItemFromGroup**

Description:

Method	Return values	Description
<b>deleteItemFromGroup(String uuid, long groupId)</b>	<b>void</b>	Remove a node from a related group.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (RelationGroup rg : ws.relation.getRelationGroups("930baee0-8712-41b0-85c0-81d21e55742f")) {
                if (rg.getName().equals("staple")) {
                    ws.relation.deleteItemFromGroup("930baee0-8712-41b0-85c0-81d21e55742f", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Report samples

### Basics

The table below shows how should be passed variables based on field type:

Field type	Type	Description
<b>Date</b>	<b>String</b>	Use the pattern yyyy-MM-dd ( year - month - day ) 2018-10-04
<b>Select multiple</b>	<b>String</b>	Use "," to split each value "value1,value2,value3"

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Report methods from "**report**" class as is shown below:

```
ws.report.getReports(true)
```

### Methods

#### getReports

Description:

Method	Return values	Description
<b>getReports(boolean active)</b>	<b>List&lt;Report&gt;</b>	Returns a list of reports.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (Report rep : ws.report.getReports(true)) {
                System.out.println(rep);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getReport**

Description:

Method	Return values	Description
<b>getReport(long rpId)</b>	<b>Report</b>	Returns a reports.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Report rep = ws.report.getReport(1);
            System.out.println(rep);
        } catch (Exception e) {
        }
    }
}

```

```

        e.printStackTrace();
    }
}

```

**generateDownloadReportToken**

Description:

Method	Return values	Description
<b>generateDownloadReportToken(long rpId)</b>	<b>String</b>	Return the token for downloading the report.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String token = ws.report.generateDownloadReportToken(1);
            System.out.println(token);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**execute**

Description:

Method	Return values	Description
<b>execute(long rpId, Map&lt;String, String&gt; params, String format, String uuid)</b>	<b>InputStream</b>	Return a document result of executing a report.

**i** Available formats:

- Report.FORMAT\_CSV

- Report.FORMAT\_DOCX
- Report.FORMAT\_HTML
- Report.FORMAT\_ODT
- Report.FORMAT\_PDF
- Report.FORMAT\_RTF
- Report.FORMAT\_TEXT

The parameter **uuid** is the UUID of a node ( this parameter is optional, usually has sense with reports executed from user interface where the results of the reports depend on the node selected ).

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> params = new HashMap<>();
            params.put("from_date", "2021-01-01");
            params.put("to_date", "2021-05-01");

            long rpld = 1;
            String format = Report.FORMAT_PDF;
            String dstld = "c5c87bd3-0c03-4bba-ab2f-7184c23a26d8";
            String uuid = "";
            InputStream is = ws.report.execute(rpld, params, format, dstld, uuid);
            OutputStream fos = new FileOutputStream("/home/openkm/report/document Create.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### executeSql

Description:

Method	Return values	Description
<b>executeSql(long rpId)</b>	<b>InputStream</b>	Return a document result of executing a report sql.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
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);

            // executeReportSql
            long rpId = 1;
            InputStream is = ws.report.executeSql(rpId);
            OutputStream fos = new FileOutputStream("/home/openkm/report/activity.csv");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
    
```

**save**

Description:

Method	Return values	Description
<b>save(long rpId, Map&lt;String, String&gt; params, String format, String dstId, docName, String uuid)</b>	Document	Execute the report and save the resulting document.

 Available formats:

- Report.FORMAT\_CSV
- Report.FORMAT\_DOCX
- Report.FORMAT\_HTML
- Report.FORMAT\_ODT
- Report.FORMAT\_PDF
- Report.FORMAT\_RTF
- Report.FORMAT\_TEXT

The values of the **dstld** parameter should be a folder or record UUID.

The parameter **docName** is the file name of the report.

The parameter **uuid** is the UUID of a node ( this parameter is optional, usually has sense with reports executed from user interface where the results of the reports depend on the node selected ).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.HashMap;
import java.util.Map;

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);
            Map<String, String> params = new HashMap<>();
            params.put("from_date", "2021-01-01");
            params.put("to_date", "2021-05-01");

            long rpld = 1;
            String format = Report.FORMAT_PDF;
            String dstld = "c5c87bd3-0c03-4bba-ab2f-7184c23a26d8";
            String docName = "Document create.pdf";
            String uuid = "";
            Document documenReport = ws.report.save(rpld, params, format, dstld, docName, uuid);
            System.out.println(documenReport);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**saveSql**

Description:

Method	Return values	Description
<b>saveSql(long rpId, Map&lt;String, String&gt; params, String dstId, docName)</b>	Document	Execute the report sql and save the resulting document csv

**i** The values of the **dstId** parameter should be a folder or record UUID.

The parameter **docName** is the file name of the report.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.HashMap;
import java.util.Map;

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);

            // saveSql
            long rpId = 1;
            String dstId = "8f64d889-1600-4cbc-8245-de6fb62214eb";
            String docName = "activity.csv";
            Document documenReport = ws.report.saveSql(rpId, new HashMap<>(), dstId, docName);
            System.out.println(documenReport);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
    
```

**getSqlReports**

Description:

Method	Return values	Description
<b>getSqlReports(boolean active)</b>	<b>List&lt;Report&gt;</b>	Returns a list of sql reports.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlReport;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);

            // getSqlReports
            List<SqlReport> sqlReports = ws.report.getSqlReports(true);
            for (SqlReport sqlReport : sqlReports) {
                System.out.println(sqlReport);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

## Repository samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Repository methods from "**repository**" class as is shown below:

```
ws.repository.getAppVersion();
```

### Methods

#### getRootFolder

Description:

Method	Return values	Description
<b>getRootFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:root"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    System.out.println(ws.repository.getRootFolder());
} catch (Exception e) {
    e.printStackTrace();
}
}
}


```

**getTrashFolder**

Description:

Method	Return values	Description
<b>getTrashFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:trash/{userId}"

The returned folder will be the user trash folder.


For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:trash/okmAdmin".

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getTrashFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getTrashFolderBase**

Description:

Method	Return values	Description

<b>getTrashFolderBase()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:trash"
-----------------------------	---------------	--

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getTrashFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### getTemplatesFolder

Description:

Method	Return values	Description
<b>getTemplatesFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:templates"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getTemplatesFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


```
}
}
```

**getPersonalFolder**

Description:

Method	Return values	Description
<b>getPersonalFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.

 For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:personal/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getPersonalFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getPersonalFolderBase**

Description:

Method	Return values	Description
<b>getPersonalFolderBase()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:personal"

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getPersonalFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

**getMailFolder**

Description:

Method	Return values	Description
<b>getMailFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.


For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:mail/okmAdmin".

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getMailFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
  }
}

```

**getMailFolderBase**

Description:

Method	Return values	Description
<b>getMailFolderBase()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:mail"

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getMailFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getCategoriesFolder**

Description:

Method	Return values	Description
<b>getCategoriesFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:categories"

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        System.out.println(ws.repository.getCategoriesFolder());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}


```

### purgeTrash

Description:

Method	Return values	Description
<b>purgeTrash()</b>	<b>void</b>	Definitively removes from repository all nodes to "/okm:trash/{userId}"

 For example if the method is executed by the user "okmAdmin" then the purged trash will be "/okm:trash/okmAdmin".

 When a node is purged it will only be able to be restored from a previously repository backup. The purge action removes the node definitively from the repository.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.repository.purgeTrash();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

### getUpdateMessage

Description:

Method	Return values	Description
<b>getUpdateMessage()</b>	<b>String</b>	Retrieves a message when there is a new OpenKM release.

There's an official OpenKM update message service available which is based on your local OpenKM version.

 The most common message is that a new OpenKM version has been released.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getUpdateMessage());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getRepositoryUuid

Description:

Method	Return values	Description
<b>getRepositoryUuid()</b>	<b>String</b>	Retrieves the installation unique identifier.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
    
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getRepositoryUuid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**hasNode**

Description:

Method	Return values	Description
<b>hasNode(String nodeId)</b>	<b>Boolean</b>	Returns a node that indicate if a node exists or not.
The value of the parameter nodeId can be a valid <b>UUID</b> or <b>path</b> .		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

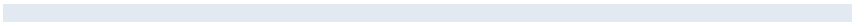
public class Test {

    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);
            System.out.println("Exists node:" + ws.repository.hasNode("373bcdd0-c082-4e7b-addd-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getNodePath**

Description:



Method	Return values	Description
<b>getNodePath(String uuid)</b>	String	Converts a node UUID to path.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getNodePath("373bccdd0-c082-4e7b-addr-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getNodeUuid

Description:

Method	Return values	Description
<b>getNodeUuid(String nodePath)</b>	String	Converts a node path to UUID.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getNodeUuid("/okm:root/test"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
  }
}

```

### getAppVersion

Description:

Method	Return values	Description
<b>getAppVersion()</b>	<b>AppVersion</b>	Returns information about OpenKM version.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getAppVersion());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### copyAttributes

Description:

Method	Return values	Description
<b>copyAttributes(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes)</b>	<b>void</b>	Copy attributes from a node to other.

The values of the dstId parameter should be a node UUID.

**i**

- 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.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.repository.copyAttributes("46762f90-82c6-4886-8d21-ad3017dd78a7", "ac9fe744-8e45-4bf4-a293-85dafr
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**executeScript**

Description:

Method	Return values	Description
<b>executeScript(InputStream is)</b>	<b>ScriptExecutionResult</b>	Executes an script.


The local script - test.bsh - used in the sample below:

```

import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

try {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    for (Folder fld : okmFolder.getChildren(null, "/okm:root")) {
        print(fld.getPath() + "\n");
    }
} catch (Exception e) {
    e.printStackTrace();
}

// Some value can also be returned as string
return "test result";
    
```

 This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ScriptExecutionResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/openkm/test.bsh");
            ScriptExecutionResult result = ws.repository.executeScript(is);
            System.out.println(result.getResult());
            System.out.println(result.getStdout());


            if (!result.getStderr().isEmpty()) {
                System.out.println("Error happened");
                System.out.println(result.getStderr());
            }

            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**executeScript**

Description:

Method	Return values	Description
<b>executeScript(String script)</b>	<b>ScriptExecutionResult</b>	Executes an script.

 This action can only be done by a super user (user with ROLE\_ADMIN).

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ScriptExecutionResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String script = "import com.openkm.util.ContextWrapper;"
                + " import org.springframework.web.context.WebApplicationContext;"
                + " import com.openkm.api.OKMFolder;"
                + " import com.openkm.bean.Folder;"
                + " WebApplicationContext cc = (WebApplicationContext) ContextWrapper.getContext();"
                + " OKMFolder okmFolder = cc.getBean(OKMFolder.class);"
                + " for (Folder fld : okmFolder.getChildren(null, \"/okm:root\")) {"
                + " print(fld.getPath());"
                + " }";
            ScriptExecutionResult result = ws.repository.executeScript(script);
            System.out.println(result.getResult());
            System.out.println(result.getStdout());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**executeSqlQuery**

Description:


Method	Return values	Description
<b>executeSqlQuery(InputStream is)</b>	<b>SqlQueryResults</b>	Executes SQL sentences.

The test.sql script used in the sample below:

```

SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;

```


The SQL script can only contains a single SQL sentence.

This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlQueryResultColumns;
import com.openkm.sdk4j.bean.SqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/test.sql");
            SqlQueryResults result = ws.repository.executeSqlQuery(is);
            for (SqlQueryResultColumns columns : result.getResults()) {
                System.out.println("UUID:" + columns.getColumns().get(0));
                System.out.println("Context:" + columns.getColumns().get(1));
                System.out.println("Name:" + columns.getColumns().get(2));
            }
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

Also the InputStream can be set as:

```

String sql = "SELECT NBS_UUID, NBS_CONTEXT, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;";
InputStream is = new ByteArrayInputStream(sql.getBytes("UTF-8"));
    
```

### executeSqlQuery

Description:

Method	Return values	Description
<b>executeSqlQuery(String sql)</b>	<b>SqlQueryResults</b>	Executes SQL sentences.

The SQL script can only contains a single SQL sentence.



This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlQueryResultColumns;
import com.openkm.sdk4j.bean.SqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            SqlQueryResults result = ws.repository.executeSqlQuery("SESELECT NBS_UUID, NBS_CONTEXT, NBS_I
            for (SqlQueryResultColumns columns : result.getResults()) {
                System.out.println("UUID:" + columns.getColumns().get(0));
                System.out.println("Context:" + columns.getColumns().get(1));
                System.out.println("Name:" + columns.getColumns().get(2));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**executeHqlQuery**

Description:

Method	Return values	Description
<b>executeHqlQuery(InputStream is)</b>	<b>HqlQueryResults</b>	Executes HQL sentences.

The test.sql script used in the sample below:

```

SELECT uuid, author from NodeBase where name = 'okm:root';
    
```



The HQL script can only contains a single HQL sentence.

This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.HqlQueryResultColumns;
import com.openkm.sdk4j.bean.HqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/test.sql");
            HqlQueryResults result = ws.repository.executeHqlQuery(is);

            for (HqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid: " + row.getColumns().get(0) + ", name: " + row.getColumns().get(1));
            }

            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Also the InputStream can be set as:

```


String sql = "SELECT uuid, name from NodeBase where name = 'okm:root'";
InputStream is = new ByteArrayInputStream(sql.getBytes("UTF-8"));

```

**executeHqlQuery**

Description:

Method	Return values	Description
<b>executeHqlQuery(String hql)</b>	<b>HqlQueryResults</b>	Executes HQL sentences.



The HQL script can only contains a single HQL sentence.

This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.HqlQueryResultColumns;
import com.openkm.sdk4j.bean.HqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            HqlQueryResults result = ws.repository.executeHqlQuery("SELECT uuid, author from NodeBase where nam
            for (HqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid: " + row.getColumns().get(0) + ", name: " + row.getColumns().get(1));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getTranslations**

Description:

Method	Return values
<b>getTranslations(String lang, String module)</b>	<b>Map&lt;String, String&gt;</b> Retrieve the 1

**i** The OpenKM translations tables can be used to retrieve actually OpenKM translations or create your own translation. By default modules values are :

- frontend ( used by default OpenKM UI ).
- extension ( used by OpenKM extension UI ).
- mobile ( used by OpenKM mobile UI ).

Example to add a new Translation module :

SQL values to be executed from [Database query](#) view:

```

DELETE FROM OKM_TRANSLATION WHERE TR_LANGUAGE='en-GB' and TR_MODULE='doc';
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
    
```

The code then should be:

```
Map<String, String> translations = ws.getTranslations("en-GB", "doc");
```

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> translations = ws.repository.getTranslations("en-GB", "frontend");
            for (String key : translations.keySet()) {
                System.out.println("key:" + key + ", with translation:" + translations.get(key) + "");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### getConfiguration

Description:

Method	Return values	Description
<b>getConfiguration(String key)</b>	<b>Configuration</b>	Retrieve the value of a configuration parameter.



If your OpenKM version have the configuration parameter named "**webservices.visible.properties**", will be restricted for non Administrator users what parameters are accessible. That means any non Administrator use who will try accessing across the webservices to configuration parameters not set into the list of values of "**webservices.visible.properties**" will get an access denied exception.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Configuration;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Configuration configuration = ws.repository.getConfiguration("system.ocr");
            System.out.println(configuration);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getChangeLog**

Description:

Method	Return values	Description
<b>getChangeLog(String nodePath, Calendar modificationsFrom)</b>	<b>List&lt;ChangeLogged&gt;</b>	Return the list of changes in some path and subfolders.



- The method is used by desktop synchronization application for retrieving the changes.

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ChangeLogged;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Calendar now = Calendar.getInstance();
            for (ChangeLogged cl : ws.repository.getChangeLog("/okm:root/synchronized", now)) {
                System.out.println(cl);
            }
        }
    }
}
```

```


    } catch (Exception e) {
        e.printStackTrace();
    }
}
}


```

**getServerTime()**

Description:

Method	Return values	Description
<b>getServerTime()</b>	<b>String</b>	Return the current server time.

 The server time returned format is ISO8601



- The method is used by desktop synchronization application for retrieving the changes

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            System.out.println(ws.repository.getServerTime());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getAvailableLocales**

Description:

Method	Return values	Description
<b>getAvailableLocales(String locale)</b>	<b>Map&lt;String, String&gt;</b>	Return the available languages

Example:

```

package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, String> locales = ws.repository.getAvailableLocales("en-GB");
            for (String key : locales.keySet()) {
                System.out.println("Language:" + key + "," + locales.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getLicenseInfo()

Description:

Method	Return values	Description
<b>getLicenseInfo()</b>	<b>LicenseInfo</b>	Return license information.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.LicenseInfo;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            LicenseInfo licenseInfo = ws.repository.getLicenseInfo();
            System.out.println(licenseInfo);
        }
    }
}

```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getClusterUuid()**

Description:

Method	Return values	Description
<b>getClusterUuid()</b>	String	Return cluster UUID.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String clusterUuid = ws.repository.getClusterUuid();
            System.out.println(clusterUuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getIsFilePlan()**

Description:

Method	Return values	Description
<b>getIsFilePlan()</b>	boolean	True when the filePlan mode is active

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

```

```
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);  
            System.out.println("FilePlan: " + ws.repository.getIsFilePlan());  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```


## Search samples

### Basics


Almosts all methods use QueryParams. Here there're some tips about how using it.

Variables	Type	Allow wildcards	Restrictions
<b>domain</b>	<b>long</b>	No.	<p>Available values:</p> <ul style="list-style-type: none"> <li>• QueryParams.DOCUMENT</li> <li>• QueryParams.FOLDER</li> <li>• QueryParams.MAIL</li> <li>• QueryParams.RECORD</li> </ul> <p>By default the value is set to QueryParams.DOCUMENT.</p> <p>For searching documents and folders use value:</p> <div style="border: 1px dashed blue; padding: 5px; display: inline-block;">                     (QueryParams.DOCUMENT   QueryParams.FOLDER)                 </div>
<b>author</b>	<b>String</b>	No.	Value must be a valid userId.
<b>name</b>	<b>String</b>	Yes.	
<b>title</b>	<b>String</b>	Yes.	
<b>keywords</b>	<b>Set&lt;String&gt;</b>	Yes.	
<b>categories</b>	<b>Set&lt;String&gt;</b>	No.	Values should be a category UUID, not use path value.

<b>content</b>		Yes.	
<b>contentType</b>		No.	Value should be a valid and registered MIME type.  Only can be applied to documents node.
<b>language</b>		No.	Value should be a valid language.  Only can be applied to documents node.
<b>folder</b>		No.	When empty is used by default "/okm:root" node.  Value should be a valid UUID, not use a path value.
<b>folderRecursive</b>	<b>Boolean</b>	No.	Only has sense to set this variable to true when the variable folder is not
<b>lastModifiedFrom</b>	<b>Calendar</b>	No.	
<b>lastModifiedTo</b>	<b>Calendar</b>	No.	
<b>mailSubject</b>	<b>String</b>	Yes.	Only apply to mail nodes.

<b>mailFrom</b>	<b>String</b>	Yes.	Only apply to mail nodes.
<b>mailTo</b>		Yes.	Only apply to mail nodes.
<b>notes</b>		Yes.	
<b>properties</b>	<b>Map&lt;String, String&gt;</b>	Yes on almost.	<p>On metadata field values like "date" can not be applied wilcards.</p> <p>The map of the properties is composed of pairs: (<code>'metadata_field_name','metada_field_value'</code>)</p> <p>For example:</p> <pre>Map&lt;String, String&gt; properties = new HashMap(); properties.put("okp:consulting.name", "name value");</pre> <p><b>Filtering by range of dates:</b></p> <pre>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&lt;String,String&gt; properties = new HashMap&lt;&gt;(); properties.put("okp:consulting.date", ISO8601.formatBasic(from</pre> <div style="border: 1px dashed orange; padding: 5px; margin: 10px 0;">  When filtering by range of dates you must set both values ( will be ignored from the OpenKM side.         </div> <p><b>To filtering by a metadata field of type multiple like this:</b></p> <pre>&lt;select label="Multiple" name="okp:consulting.multiple" type=" &lt;option label="One" value="one"/&gt; &lt;option label="Two" value="two"/&gt; &lt;option label="Three" value="three" /&gt;</pre>

			<pre>&lt;/select&gt;</pre> <p>You should do:</p> <pre>properties.put("okp:consulting.multiple", "one;two");</pre> <p>Where <b>"one"</b> and <b>"two"</b> are valid values and character <b>";"</b> is used as sep</p>
--	--	--	---

 The search operation is done only by AND logic.

Wildcard examples:

Variable	Example	Description
<b>name</b>	<b>test*.html</b>	Any document that starts with characters "test" and ends with characters ".html"
<b>name</b>	<b>test?.html</b>	Any document that starts with characters "test" followed by a single character and ends with characters ".html"
<b>name</b>	<b>?test*</b>	Any of the documents where the first character doesn't matter, but is followed by the characters, "test".


**Suggested code sample**

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method **"login"**. You can access the **"login"** method from webservice object **"ws"** as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Search methods from **"search"** class as is shown below:


```
ws.search.find(qParams, null)
```


## Methods

### find

Description:

Method	Return values	Description
<b>find(QueryParams queryParams, String propertiesPlugin)</b>	<b>List&lt;QueryResult&gt;</b>	Returns a list of results filtered by the values of the queryParams parameter.

 The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

 Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");


            for (QueryResult qr : ws.search.find(params, null)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**find**


Description:

Method	Return values	Description
<b>find(QueryParams queryParams, String sortField, boolean sortReverse, String propertiesPlugin)</b>	<b>List&lt;QueryResult&gt;</b>	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.

 Available **sortField** values:

SearchSortField.NAME  
SearchSortField.AUTHOR  
SearchSortField.LAST\_MODIFIED

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");

            for (QueryResult qr : ws.search.find(params, SearchSortField.LAST_MODIFIED, true, null)) {
                System.out.println(qr);
            }
        }
    }
}

```

```


    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}


```

**findSimpleNodeBasePaginated**


Description:

Method	Return values	Description
<b>findSimpleNodeBasePaginated(QueryParams queryParams, int offset, int limit)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

 Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods ( these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enough. Specially when the limit is a higher value you will appreciate the difference in the performance.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");
            SimpleNodeBaseResultSet rs = ws.search.findSimpleNodeBasePaginated(params, 20, 10);
            System.out.println("Total results: " + rs.getTotal());


            for (SimpleNodeBase sn : rs.getResults()) {
                System.out.println(sn);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### findSimpleNodeBasePaginated

Description:

Method	Return values	Description
<b>findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

 Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods ( these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enough. Specially when the limit is a higher value you will appreciate the difference in the performance.

 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.



Available `sortField` values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```



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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");

            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBasePaginated(params, SearchSortField.AUTHOR);
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


```
}


```

**findSimpleNodeBasePaginated**


Description:

Method	Return values	Description
<b>findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit, String pluginName)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.


 Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods ( these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enough. Specially when the limit is a higher value you will appreciate the difference in the performance.

 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.

 Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");
            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBasePaginated(params, SearchSortField.AU
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getCategorizedDocuments**

Description:

Method	Return values	Description
<b>getCategorizedDocuments(String categoryId)</b>	<b>List&lt;Document&gt;</b>	Retrieves a list of all documents related with a category.

The values of the categoryId parameter should be a category folder UUID.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;
    
```

```

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);
            for (Document doc : ws.search.getCategorizedDocuments("58c9b25f-d83e-4006-bd78-e26d7c6fb648")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**saveSearch**

Description:

Method	Return values	Description
<b>saveSearch(QueryParams params)</b>	<b>Long</b>	Saves a search parameters.
The variable queryName of the parameter params, should have to be initialized.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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 qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");
            for (QueryResult qr : ws.find(qParams, null)) {
                System.out.println(qr);
            }
            // Save the search to be used later
            qParams.setQueryName("sample search");
            ws.search.saveSearch(qParams);
        } catch (Exception e) {
        }
    }
}

```

```

        e.printStackTrace();
    }
}

```

### updateSearch

Description:

Method	Return values	Description
<b>updateSearch(QueryParams params)</b>	<b>void</b>	Updates a previously saved search parameters.

 Only can be updated as a saved search created by the same user user who's executing the method.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (QueryParams qParams : ws.getAllSearchs()) {
                if (qParams.getQueryName().equals("sample search")) {
                    // Change some value.
                    qParams.setName("admin*.html");
                    ws.search.updateSearch(qParams);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getSearch

Description:

Method	Return values	Description

<b>getSearch(int qpId)</b>	<b>QueryParams</b>	Gets saved search parameters.
 Only can be retrieved as a saved search created by the same user who's executing the method.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;


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);
            int qpId = 1; // Some valid search id
            QueryParams qParams = ws.search.getSearch(qpId);
            System.out.println(qParams);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getAllSearches**

Description:

Method	Return values	Description
<b>getAllSearches()</b>	<b>List&lt;QueryParams&gt;</b>	Retrieves a list of all saved search parameters.
 Only will be retrieved the list of the saved searches created by the same user who's executing the method.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;
    
```


```
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);
            for (QueryParams qParams : ws.search.getAllSearches()) {
                System.out.println(qParams);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**deleteSearch**

Description:

Method	Return values	Description
<b>deleteSearch(int qpId)</b>	<b>void</b>	Deletes a saved search parameters.

 Only can be deleted as a saved search created by the same user user who's executing the method.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int qpId = 1; // Some valid search id
            ws.search.deleteSearch(qpId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### getSearchConfig

Description:

Method	Return values	Description
<b>getSearchConfig(String pluginName)</b>	<b>NodeSearchConfig</b>	Return a NodeSearchConfig object.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeSearchConfig;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            NodeSearchConfig nodeSearchConfig = ws.search.getSearchConfig("com.openkm.plugin.search.TestNode");
            System.out.println(nodeSearchConfig);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getSearchPlugins

Description:

Method	Return values	Description
<b>getSearchPlugins()</b>	<b>SearchPluginList</b>	Return a SearchPluginList object.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SearchPlugin;
import com.openkm.sdk4j.bean.SearchPluginList;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

        SearchPluginList pluginList = ws.search.getSearchPlugins();
        for (SearchPlugin searchPlugin : pluginList.getSearchPlugins()) {
            System.out.println(searchPlugin);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

### findAllDefaultByNodeClass

Description:

Method	Return values	Description
<b>findAllDefaultByNodeClass(long nclId)</b>	<b>List&lt;QueryParams&gt;</b>	Retrieve a list of saved searches of a NodeClass

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long nclId = 1;
            List<QueryParams> results = ws.search.findAllDefaultByNodeClass(nclId);
            for (QueryParams params : results) {
                System.out.println(params);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

## findByQuery

Description:

Method	Return values	Description
<b>findByQuery(String query, String propertiesPlugin)</b>	<b>List&lt;QueryResult&gt;</b>	Returns a list of results filtered by the query parameter.

**i** 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](#).

**✓** The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);
            List<QueryResult> results = ws.search.findByQuery("keyword:test AND name:t*.pdf", null);
            for (QueryResult qr : results) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

## findByQuery

Description:



Method	Return values	Description
<b>findByQuery(String query, String sortField, boolean sortReverse, String propertiesPlugin)</b>	<b>List&lt;QueryResult&gt;</b>	Returns a list of results filtered by the query parameter.

**i** 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](#).

**i** Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

**✓** The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);
            List<QueryResult> results = ws.search.findByQuery("keyword:test AND name:t*.pdf", SearchSortField.NAME);
            for (QueryResult qr : results) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### findSimpleNodeBaseByQueryPaginated

Description:

Method	Return values	Description
<b>findSimpleNodeBaseByQueryPaginated(String query, int offset, int limit)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of paginated results filtered by the values of the query parameter.

**i** 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](#).

**✓** 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.

**i** 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND name:" + name);
    for (SimpleNodeBase nodeBase : result.getResults()) {
        if (nodeBase != null) {
            System.out.println(nodeBase.toString());
        }
    }
    System.out.println("Total: " + result.getTotal());
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**findSimpleNodeBaseByQueryPaginated**

Description:

Method	Return values	Description
<b>findSimpleNodeBaseByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit)</b>	<b>SimpleNodeBaseResultSet</b>	Returns a list of paginated results filtered by the values of the query parameter.

**i** 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](#).

**i** Available sortField values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

**✓** 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

**i** 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND name");
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```


**findWithMetadata**

Description:

Method	Return values	Description
<b>findWithMetadata(QueryParams queryParams, String propertiesPlugin, List&lt;String&gt; groups)</b>	<b>List&lt;QueryResult&gt;</b>	Returns a list of results with metadata values filtered by the queryParams parameter.

 • The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

• The parameter "groups" must be valid metadata group names.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

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 qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");




            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");

            for (QueryResult qr : ws.search.findWithMetadata(qParams, null, groups)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**findWithMetadata**

Description:

Method	Return values	Description
<b>findWithMetadata(QueryParams queryParams, String</b>		Returns a list of results

<b>sortField, boolean sortReverse, String propertiesPlugin, List&lt;String&gt; groups)</b>	<b>List&lt;QueryResult&gt;</b>	with metadata values filtered by the queryParams parameter.
<div style="border: 1px dashed green; padding: 10px;">  <ul style="list-style-type: none"> <li>• The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.</li> <li>• The parameter "groups" must be valid metadata group names.</li> </ul> </div>		
<div style="border: 1px dashed blue; padding: 10px;">  Available sortField values: <div style="border: 1px dashed blue; padding: 5px; margin-top: 5px;"> SearchSortField.NAME  SearchSortField.AUTHOR  SearchSortField.LAST_MODIFIED </div> </div>		
<div style="border: 1px dashed blue; padding: 10px;">  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. </div>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

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*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");
            for (QueryResult qr : ws.search.findWithMetadata(params, SearchSortField.NAME, true, null, groups)) {
                System.out.println(qr);
            }
        }
    }
}

```

```


    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**findWithMedataPaginated**

Description:


Method	Return values	Description
<b>findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List&lt;String&gt; groups)</b>	<b>ResultSet</b>	Returns a list of paginated results with metadata values filtered by 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" says 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.

The parameter "groups" must be valid metadata group names.

 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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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 qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");

            ResultSet rs = ws.search.findWithMetadataPaginated(qParams, 0, 10, null, groups);
            System.out.println("Total results: " + rs.getTotal());


            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**findWithMedataPaginated**

Description:

Method	Return values	Description
<b>findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List&lt;String&gt; groups)</b>	<b>ResultSet</b>	Returns a list of paginated results with metadata values filtered by 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" says 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.

The parameter "groups" must be valid metadata group names.



Available `sortField` values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```



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.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

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*");

            ResultSet rs = ws.search.findWithMetadataPaginated(params, SearchSortField.AUTHOR, true, 0, 10, null, 0);
            System.out.println("Total results: " + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
```

```

        System.out.println(qr);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### getMimeType

Description:

Method	Return values	Description
<b>getMimeType()</b>	<b>List&lt;MimeType&gt;</b>	Retrieves a list of mimetypes.

 Only will be retrieved the list of the mimeTypees what may be used in the search.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MimeType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (MimeType mimeType : ws.search.getMimeTypes()) {
                System.out.println(mimeType);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


### csvExport

Description:

Method	Return values	Description

<b>csvExport(String token, String lang, QueryParams queryParams, boolean compact)</b>	<b>InputStream</b>	Export as a csv a list of results filtered by the values of the queryParams parameter.
---	--------------------	--

The parameter **lang** must be ISO 691-1 compliant.

 More information at: [https://en.wikipedia.org/wiki/List\\_of\\_ISO\\_639-1\\_codes](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.apache.commons.io.IOUtils;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            InputStream is = ws.search.csvExport("es-ES", params, false);
            OutputStream fos = new FileOutputStream("/home/openkm/okm/export.csv");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Shard samples

### Basics



Example of uuid:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Shard methods from "**shard**" class as is shown below:

```
ws.shard.getShards()
```

### Methods

#### getShards

Description:

Method	Return values	Description
<b>getShards()</b>	<b>List&lt;Shard&gt;</b>	Returns a list of all shards

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Shard;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            for (Shard shard : ws.shard.getShards()) {
                System.out.println(shard);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**setCurrentShard**

Description:

Method	Return values	Description
<b>setCurrentShard(long shardId)</b>	<b>void</b>	Change the current shard.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            long shardId = 1;
            ws.shard.setCurrentShard(1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**changeShard**

Description:

Method	Return values	Description
<b>changeShard(String uuid, long shardId)</b>	<b>void</b>	Change the shardId of the uuid

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            long shardId = 1;
            ws.shard.changeShard("e2391b01-ce10-42c7-94a9-b0d6b394048e", shardId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Stamp samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Stamp methods from "**stamp**" class as is shown below:

```
ws.stamp.getAllStamps();
```

### Methods

#### getAllStamps

Description:

Method	Return values	Description
<code>getAllStamps()</code>	<code>List&lt;StampItem&gt;</code>	Returns a list of the stamp items

Example:

```
package com.openkm;  
  
import java.util.List;  
  
import com.openkm.sdk4j.OKMWebservicesFactory;  
import com.openkm.sdk4j.bean.StampItem;  
import com.openkm.sdk4j.impl.OKMWebservices;  
  
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);
    List<StampItem> result = ws.stamp.getAllStamps();
    for (StampItem stampItem : result) {
        System.out.println(stampItem.getName());
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**getStampTextByPk**

Description:

Method	Return values	Description
<b>getStampTextByPk(long id, String uuid)</b>	<b>StampText</b>	Return the stamp of type Text by Id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampText;
import com.openkm.sdk4j.impl.OKMWebservices;

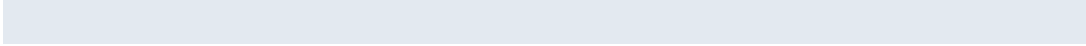
public class Test {

    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);
            int stampTextId = 1;
            StampText stampText = ws.stamp.getStampTextByPk(stampTextId, "928de61e-6ceb-4f94-bf20-9ba2405c5c");
            System.out.println(stampText);
            System.out.println(stampText.getName());
            System.out.println(stampText.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getStampImageByPk**

Description:



Method	Return values	Description
<b>getStampImageByPk(long id, String uuid)</b>	<b>StampImage</b>	Return the stamp of type image by Id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampImage;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            int stampImageId = 3;
            StampImage stampImage = ws.stamp.getStampImageByPk(stampImageId, "928de61e-6ceb-4f94-bf20-9ba2");
            System.out.println(stampImage);
            System.out.println(stampImage.getName());
            System.out.println(stampImage.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getStampBarcodeByPk**

Description:

Method	Return values	Description
<b>getStampBarcodeByPk(long id, String uuid)</b>	<b>StampBarcode</b>	Return the stamp of type barcode by Id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampBarcode;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    int stampBarcodeId = 3;
    StampBarcode stampBarcode = ws.stamp.getStampBarcodeByPk(stampBarcodeId, "6330d2a0-529f-4c14-t
System.out.println(stampBarcode);
    System.out.println(stampBarcode.getName());
    System.out.println(stampBarcode.getDescription());
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**calculateStampCoordinates**

Description:

Method	Return values	Description
<b>calculateStampCoordinates(long imgToStampWidth, long imgToStampHeight, long floatingDivWidth, long floatingDivHeight, String exprX, String exprY, String stampType, String stampAlign)</b>	<b>StampCoordinates</b>	Return the calculated stamp coordinates.

**i** In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE\_WIDTH
- IMAGE\_HEIGHT
- PAGE\_WIDTH
- PAGE\_HEIGHT
- PAGE\_CENTER
- PAGE\_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

The parameter **stampAlign**.

**i** Available values:

- text

- [image](#)

The parameter **stampAlign** is the Text Align

**i** Available values:

- left
- center
- right

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampCoordinates;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingDivWidth = 250;
            long floatingDivHeight = 300;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            String stampType = "text";
            String stampAlign = "center";
            StampCoordinates stampCoordinates = ws.stamp.calculateStampCoordinates(imgToStampWidth, imgToStampHeight,
                floatingDivWidth, floatingDivHeight, exprX, exprY, stampType, stampAlign);
            System.out.println("X = " + stampCoordinates.getX() + ", Y = " + stampCoordinates.getY());

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**calculateStampExpressions**

Description:

Method	Return values	Description

<p><b>calculateStampExpressions(long imgToStampWidth, long imgToStampHeight, long posX, long posY)</b></p>	<p><b>StampExpressions</b></p>	<p>Return the calculated stamp expressions for X and Y.</p>
--	--------------------------------	---

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampExpressions;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long posX = 10;
            long posY = 50;
            StampExpressions stampExpressions = ws.stamp.calculateStampExpressions(imgToStampWidth, imgToStampHeight, posX, posY);
            System.out.println("X = " + stampExpressions.getExprX() + ", Y = " + stampExpressions.getExprY());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**stampText**

Description:

Method	Return values	Description
<b>stampText(String uuid, long id)</b>	<b>void</b>	Stamp text in a document.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    long stampTextId = 1;
    ws.stamp.stampText("babe24df-c443-49a5-9453-39dfc9b27924", stampTextId);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**stampImage**

Description:

Method	Return values	Description
<b>stampImage(String uuid, long id)</b>	<b>void</b>	Stamp image in a document.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stampImageId = 1;
            ws.stamp.stampImage("babe24df-c443-49a5-9453-39dfc9b27924", stampImageId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**stampBarcode**

Description:

Method	Return values	Description
<b>stampBarcode(String uuid, long id)</b>	<b>void</b>	Stamp barcode in a document.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stampBarcodeId = 1;
            ws.stamp.stampBarcode("f02053e3-2264-4040-bb84-67983a0208f7", stampBarcodeId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**stampImageManually**

Description:

Method	Return values	Description
<b>stampImageManually(String uuid, long id, String exprX, String exprY, String range, String personalStampUuid)</b>	<b>void</b>	Stamp in a document with an image that is in openkm

**i** The **id** is the id of an image stamp which is created in the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

The **personalStampUuid** must be a valid document **UUID**. The document must be an image ( jpg or png ).

**i** In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE\_WIDTH
- IMAGE\_HEIGHT
- PAGE\_WIDTH
- PAGE\_HEIGHT
- PAGE\_CENTER

• PAGE\_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long id = 1;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            ws.stamp.stampImageManually("055b5206-35d0-4351-ba92-c304bbba7ddf", id, exprX, exprY, "", "ac9cdfb4");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**stampTextCustom**

Description:

Method	Return values	Description
<b>stampTextCustom(String uuid, long id, String text, String exprX, String exprY, String range)</b>	<b>void</b>	Stamp in a document with a custom text.

**i** The **id** is the id of an image stamp which is created in the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.



In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE\_WIDTH
- IMAGE\_HEIGHT
- PAGE\_WIDTH
- PAGE\_HEIGHT
- PAGE\_CENTER
- PAGE\_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.io.FileInputStream;
import java.io.InputStream;

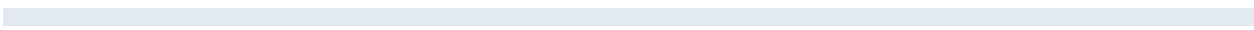
public class Test {

    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);
            long stampTextId = 1;
            String text = "OpenKM";
            String exprX = "PAGE_CENTER";
            String exprY = "PAGE_MIDDLE";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            ws.stamp.stampTextCustom("05b14eee5-2e57-4d1d-925a-c9bc3f526e24", stampTextId, text, exprX, exprY,
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**stampImageCustom**

Description:



Method	Return values	Description				
<b>stampImageCustom(String uuid, long id, String exprX, String exprY, String range, double ratio, InputStream is)</b>	<b>void</b>	Stamp in a document with a custom image.				
<div style="border: 1px dashed #ccc; padding: 10px; background-color: #e6f2ff;"> <p><b>i</b> The <b>id</b> is the id of an image stamp which is created in the administrator.</p> <p>The <b>exprX</b> is the expression to set the position in X coordinates.</p> <p>The <b>exprY</b> is the expression to set the position in Y coordinates.</p> </div>						
<div style="border: 1px dashed #ccc; padding: 10px; background-color: #e6f2ff;"> <p><b>i</b> In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:</p> <ul style="list-style-type: none"> <li>• IMAGE_WIDTH</li> <li>• IMAGE_HEIGHT</li> <li>• PAGE_WIDTH</li> <li>• PAGE_HEIGHT</li> <li>• PAGE_CENTER</li> <li>• PAGE_MIDDLE</li> </ul> <p>So to center a stamp in the page you can use:</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="padding: 5px;">Expr. X</td> <td style="padding: 5px; text-align: center;">PAGE_CENTER</td> </tr> <tr> <td style="padding: 5px;">Expr. Y</td> <td style="padding: 5px; text-align: center;">PAGE_MIDDLE</td> </tr> </table> </div>			Expr. X	PAGE_CENTER	Expr. Y	PAGE_MIDDLE
Expr. X	PAGE_CENTER					
Expr. Y	PAGE_MIDDLE					

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.io.FileInputStream;
import java.io.InputStream;

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);
        }
    }
}
    
```

```

        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");
        ws.stamp.stampImageCustom("055b5206-35d0-4351-ba92-c304bbba7ddf", id, exprX, exprY, "", 0, is);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

**calculateFlyImageDimensions**

Description:

Method	Return values	Description
<b>calculateFlyImageDimensions(String uuid, long imgToStampWidth, long imgToStampHeight, long floatingImageWidth, long floatingImageHeight, int pageNumber)</b>	<b>StampImageSize</b>	Return the calculated image to stamp height and width size.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampImageSize;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingImageWidth = 300;
            long floatingImageHeight = 300;
            int pageNumber = 2;

            StampImageSize stampImageSize = ws.stamp.calculateFlyImageDimensions("babe24df-c443-49a5-9453-39
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

**getPersonalStamps**

Description:

Method	Return values	Description
<b>getPersonalStamps()</b>	<b>List&lt;Document&gt;</b>	Returns a list of personal seals.

**i** Return a list of documents of type image ( jpg or png ) from /okm:templates/ userId/stamps

Example:

```

package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            List<Document> result = ws.stamp.getPersonalStamps();
            for (Document doc : result) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getPersonalStampImage**

Description:

Method	Return values	Description
<b>getPersonalStampImage(String uuid, long id)</b>	<b>StampPersonalImage</b>	Return the personal seal processed.

**i** When stamping a document with an image, the image also can have a text layer.  
This method processes the image and returns an image with the text layer.

The **uuid** must be an **uuid** returned by the method `getPersonalStamps`.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampPersonallImage;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long stampId = 1;
            StampPersonallImage stampPersonallImage = ws.stamp.getPersonalStampImage("928de61e-6ceb-4f94-bf21");
            System.out.println(stampPersonallImage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Task samples

### Basics


Task fields description:

Field	Type	Description	Mandatory
<b>Id</b>	<b>Long</b>	Internal task id.  <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;">  It is automatically set by application.                 </div>	Not applicable.
<b>Owner</b>	<b>String</b>	It contains the OpenKM userId.  <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;">  It is automatically set by application.                 </div>	Not applicable.
<b>Subject</b>	<b>String</b>	The topic of the task.	Yes.
<b>Description</b>	<b>String</b>	The description of the task.	No.
<b>Start</b>	<b>Calendar</b>	When the task might start.	Yes.
<b>End</b>	<b>Calendar</b>	When the task might ending.	No.
<b>Status</b>	<b>TaskStatus</b>	The task status.  <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;">  You administering the list of status values.                 </div>	Yes.
<b>Project</b>	<b>TaskProject</b>	The project related with the task.  <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;">  You administering the list of project values.                 </div>	Yes.

<b>Type</b>	<b>TaskType</b>	<p>The task type.</p> <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px;">  You administering the list of task type values.         </div>	Yes.
<b>Progress</b>	<b>Integer</b>	<p>The numeric progress status.</p> <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px;">  The range of allowed values is from 0 to 100. Where 100% indicates a completed task.         </div>	Yes.
<b>RepeatGroup</b>	<b>Long</b>	<p>When a task is repeated along time is a member of a group. The group is identified by an unique repeat group id.</p>	No.
<b>ReminderStartValue</b>	<b>Integer</b>	<p>How many days before the task starting, the system might send a mail notification to the user.</p>	No.
<b>ReminderEndValue</b>	<b>Integer</b>	<p>How many days before the task ending, the system might send a mail notification to the user.</p>	No.
<b>ReminderStartUnit</b>	<b>String</b>	<p>Reminder start units.</p> <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px;">  Allowed values:           <ul style="list-style-type: none"> <li>"m" for minutes.</li> <li>"h" for hours.</li> <li>"d" for days.</li> </ul> </div>	Mandatory when <b>ReminderStartValue</b> greater than 0.
<b>ReminderEndUnit</b>	<b>String</b>	<p>Reminder end units.</p> <div style="border: 1px dashed blue; padding: 5px; margin-top: 10px;">  Allowed values:           <ul style="list-style-type: none"> <li>"m" for minutes.</li> </ul> </div>	Mandatory when <b>ReminderEndValue</b> greater than 0.

		<ul style="list-style-type: none"> <li>• "h" for hours.</li> <li>• "d" for days.</li> </ul>	
<b>Users</b>	<b>Set&lt;String&gt;</b>	Collection of assigned users to the task.	No. But At least should be a user or role assigned.
<b>Roles</b>	<b>Set&lt;String&gt;</b>	Collection of assigned roles to the task.	No. But At least should be a user or role assigned.
<b>Documents</b>	<b>Set&lt;String&gt;</b>	List of UUID's of the related documents.	No.
<b>Folders</b>	<b>Set&lt;String&gt;</b>	List of UUID's of the related folders.	No.
<b>Mails</b>	<b>Set&lt;String&gt;</b>	List of UUID's of the related mails.	No.
<b>Records</b>	<b>Set&lt;String&gt;</b>	List of UUID's of the related records.	No.

On all methods, you'll see parameter named "**token**". When accessing application across SOAP the login process returns a token, what is used to identify the user on all the exposed methods. From default application execution context you must use the "**null**" value what indicates the application must use the "**user session**".

 In special cases, you might be "**promoted as Administrator**" using the "**administrator token**".

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```


**Suggested code sample**

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

 Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Task methods from " **task**" class as is shown below:


```
ws.task.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, subject);
```

## Methods


### getAssigned

Description:

Method	Return values	Description
<b>getAssigned(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)</b>	<b>List&lt;Task&gt;</b>	Retrieve a list of tasks assigned to a user, filtered and paginated.

 Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optinal)
- The subject parameter the topic of the task.

 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



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

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);

            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "");
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**getActive**

Description:

Method	Return values	Description
<b>getActive(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)</b>	<b>List&lt;Task&gt;</b>	Retrieve a list of active tasks assigned to a user, filtered and paginated.

**i** Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optional)
- The subject parameter the topic of the task.

**✓** 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.

**i** 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 parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

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);

            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getActive(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "");
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getFinished**

Description:

Method	Return values	Description
<b>getFinished(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)</b>	<b>List&lt;Task&gt;</b>	Retrieve a list of finished tasks assigned to a user, filtered and paginated.



#### Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optional)
- The subject parameter the topic of the task.



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



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;  
  
import com.openkm.sdk4j.OKMWebservicesFactory;  
import com.openkm.sdk4j.bean.*;  
import com.openkm.sdk4j.impl.OKMWebservices;
```

```

import java.util.Calendar;

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);

            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getFinished(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "")
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getNotified**

Description:

Method	Return values	Description
<b>getNotified(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)</b>	<b>List&lt;Task&gt;</b>	Retrieve a list of notified tasks assigned to a user, filtered and paginated.

**i** Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).

- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter optional)
- The to date to filter (this parameter optional)
- The subject parameter the topic of the task.



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



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

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);

            String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);
```

```

Calendar to = Calendar.getInstance();
to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

long statusId = 1; // A valid task status id
long projectId = 1; // A valid project id
long typeId = 1; // A valid type id

TaskList listAssigned = ws.task.getNotified(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "");
for (Task task : listAssigned.getTasks()) {
    System.out.println(task);
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**getStatus**

Description:

Method	Return values	Description
<b>getStatus()</b>	<b>List&lt;TaskStatus&gt;</b>	Retrieve a list of all the task status.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (TaskStatus ts : ws.task.getStatus()) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProjects**

Description:

Method	Return values	Description
--------	---------------	-------------

<b>getProjects(boolean filterActive)</b>	<b>List&lt;TaskProject&gt;</b>	Retrieve a list of all the task projects.
--	--------------------------------	---

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (TaskProject tp : ws.task.getTProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**getTypes**

Description:

Method	Return values	Description
<b>getTypes(boolean filterActive)</b>	<b>List&lt;TaskType&gt;</b>	Retrieve a list of all the task types.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (TaskType tt : ws.task.getTypes(true)) {
                System.out.println(tt);
            }
        }
    }
}
    
```

```

        System.out.println(tt);
    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getAssignedCount**

Description:

Method	Return values	Description
<b>getAssignedCount(long statusId, long projectId, long typeId)</b>	<b>long</b>	Return the number of tasks assigned to a user.

**i** Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getAssignedCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getActiveCount

Description:

Method	Return values	Description
<b>getActiveCount(long statusId, long projectId, long typeId)</b>	<b>Long</b>	Return the number of active tasks to a user.

**i** Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getActiveCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getFinishedCount

Description:

Method	Return values	Description

<b>getFinishedCount(long statusId, long projectId, long typeId)</b>	<b>Long</b>	Return the number of finished tasks to a user.
---	-------------	--

**i** Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getFinishedCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

### getNotifiedCount

Description:

Method	Return values	Description
<b>getNotifiedCount(long statusId, long projectId, long typeId)</b>	<b>Long</b>	Return the number of notified tasks assigned to a user.

**i** Filter parameters description:

- The `statusId` parameters must be a valid status id.
- The `projectId` parameter must be a valid project id.
- The `typeId` parameters must be a valid type id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getNotifiedCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**create**

Description:

Method
<p><b>create(String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List&lt;String&gt; notificationUsers, List&lt;String&gt; externalUsers, List&lt;String&gt; relatedDocuments, List&lt;String&gt; relatedFolders, List&lt;String&gt; relatedRecords, List&lt;String&gt; relatedMails, String repeatExpression, String repeatExpression, String formatDate, int repeatTimes, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)</b></p>

**i** The `repeatExpression` parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of month, or day of week) match the current time. The scheduler examines crontab entries o

minute. The time and date fields are:

```

* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ???? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same)
? ? ? ?????????? month (1 - 12)
? ? ?????????????? day of month (1 - 31)
? ?????????????????? hour (0 - 23)
???????????????????? min (0 - 59)

```

Example:

```

package com.openkm;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

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);
            String subject = "task subject";

            Calendar calendar = Calendar.getInstance();
            String start = ISO8601.formatBasic(calendar);

            calendar.add(Calendar.DATE, 15);
            String end = ISO8601.formatBasic(calendar);

            String description = "description test";

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            List<String> notificationUsers = new ArrayList<>();
            notificationUsers.add("gnujavasergio");
            notificationUsers.add("sochoa");

            List<String> externalUsers = new ArrayList<>();
            externalUsers.add("gnu.java.sergio@gmail.com");

            List<String> relatedDocuments = new ArrayList<>();
            relatedDocuments.add("d701c503-87fc-4a9e-829e-478dc375eb83");
            relatedDocuments.add("f02053e3-2264-4040-bb84-67983a0208f7");

            List<String> relatedFolders = new ArrayList<>();

```

```

relatedFolders.add("0a777b34-f518-4da7-9e58-8b1425e05add");
relatedFolders.add("271cc3aa-218e-4574-8065-c34c704f4a20");

List<String> relatedMails = new ArrayList<>();
relatedMails.add("3ae1bb26-acdf-4463-a0c1-06cc62f55b95");
relatedMails.add("c599be73-7831-4e93-be91-453baeda2b00");

List<String> relatedRecords = new ArrayList<>();
relatedRecords.add("96801b33-e92e-436b-8ed7-5d8dea904673");
relatedRecords.add("23b50095-aa18-4d16-ae75-74b994a2f2b4");

String currentDate = "31/04/2018";
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
calendar.setTime(sdf.parse(currentDate));

String repeatExpression = "0 0 1 * *";

calendar.add(Calendar.DATE, 1);
String repeatUntil = ISO8601.formatBasic(calendar);

// createTask
Task newTask = ws.task.create(subject, start, end, description, statusId, projectId, typeId,
    "okmAdmin", notificationUsers, externalUsers, relatedDocuments, relatedFolders, relatedRecords, relat
    "dd-MM-yyyy", 10, "m", 5, "m", 10);
System.out.println(newTask);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**update**

Description:

Method	Return values
<b>update(long taskId, String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List&lt;String&gt; notificationUsers, List&lt;String&gt; externalUsers, List&lt;String&gt; relatedDocuments, List&lt;String&gt; relatedFolders, List&lt;String&gt; relatedRecords, List&lt;String&gt; relatedMails, String owner, String repeatExpression, String repeatUntil, String formatDate, int repeatTimes, int progress, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)</b>	<b>Task</b>

**i** The **repeatExpression** parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of month, or day of week) match the current time. The scheduler examines crontab entries o

minute. The time and date fields are:

```

* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ???? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same)
? ? ? ?????????? month (1 - 12)
? ? ?????????????? day of month (1 - 31)
? ?????????????????? hour (0 - 23)
???????????????????? min (0 - 59)

```

Example:

```

package com.openkm;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

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);
            String subject = "update task subject";

            Calendar calendar = Calendar.getInstance();
            String start = ISO8601.formatBasic(calendar);

            calendar.add(Calendar.DATE, 15);
            String end = ISO8601.formatBasic(calendar);

            String description = "Update description test";

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            List<String> notificationUsers = new ArrayList<>();
            notificationUsers.add("gnujavasergio");
            notificationUsers.add("sochoa");

            List<String> externalUsers = new ArrayList<>();
            externalUsers.add("gnu.java.sergio@gmail.com");

            List<String> relatedDocuments = new ArrayList<>();
            relatedDocuments.add("d701c503-87fc-4a9e-829e-478dc375eb83");
            relatedDocuments.add("f02053e3-2264-4040-bb84-67983a0208f7");

            List<String> relatedFolders = new ArrayList<>();

```

```

relatedFolders.add("0a777b34-f518-4da7-9e58-8b1425e05add");
relatedFolders.add("271cc3aa-218e-4574-8065-c34c704f4a20");

List<String> relatedMails = new ArrayList<>();
relatedMails.add("3ae1bb26-acdf-4463-a0c1-06cc62f55b95");
relatedMails.add("c599be73-7831-4e93-be91-453baeda2b00");

List<String> relatedRecords = new ArrayList<>();
relatedRecords.add("96801b33-e92e-436b-8ed7-5d8dea904673");
relatedRecords.add("23b50095-aa18-4d16-ae75-74b994a2f2b4");

String currentDate = "31/04/2018";
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
calendar.setTime(sdf.parse(currentDate));

String repeatExpression = "0 0 1 * *";

calendar.add(Calendar.DATE, 1);
String repeatUntil = ISO8601.formatBasic(calendar);

long taskId = 1;
Task updateTask = ws.task.update(taskId, subject, start, end, description,
    statusId, projectId, typeId, "okmAdmin", notificationUsers, externalUsers,
    relatedDocuments, relatedFolders, relatedRecords, relatedMails,
    "okmAdmin", repeatExpression, repeatUntil, "dd-MM-yyyy",
    10, 10, "m", 5, "m", 10);
System.out.println(updateTask);
} catch (Exception e) {
    e.printStackTrace();
}
}
}
}

```

## getTask

Description:

Method	Return values	Description
<b>getTask(long taskId)</b>	<b>Task</b>	Retrieve a task.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long taskId = 2;

```

```

        Task task = ws.task.getTask(taskId);
        System.out.println(task);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**delete**

Description:

Method	Return values	Description
<b>delete(long taskId)</b>	<b>void</b>	Delete a task.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long taskId = 2;
            ws.task.delete(taskId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**createProject**

Description:

Method	Return values	Description
<b>createProject(String name, boolean active, String description)</b>	<b>TaskProject</b>	Create a new task project.


The boolean parameter "active", indicates if your project is active or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            TaskProject tp = ws.task.createProject("Project one", true, "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**updateProject**

Description:

Method	Return values	Description
<b>updateProject(long projectId, boolean active, String name, String description, )</b>	<b>TaskProject</b>	Update a task project.

 The boolean parameter "active", indicates if your project is active or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    long projectId = 4; // A valid project id
    TaskProject tp = ws.task.getTaskProject(projectId);
    System.out.println(tp);
    tp = ws.task.updateProject(projectId, false, "cancelled", "Description");
    System.out.println(tp);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

**deleteProject**

Description:

Method	Return values	Description
<b>deleteProject(long projectId)</b>	<b>void</b>	Delete a task project.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 5; // A valid project id
            ws.task.deleteProject(projectId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProject**

Description:

Method	Return values	Description
<b>getProject(long projectId)</b>	<b>TaskProject</b>	Return a task project object by id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long projectId = 1; // A valid project id
            TaskProject tp = ws.task.getProject(projectId);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProjects**

Description:

Method	Return values	Description
<b>getProjects(boolean filterActive)</b>	<b>List&lt;TaskProject&gt;</b>	Retrieve a list of all the task projects.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (TaskProject tp : ws.task.getProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```


    }
}

```

### createType

Description:

Method	Return values	Description
<b>createType(String name, boolean active, String description)</b>	<b>TaskType</b>	Create a new task type.

 The boolean parameter "active", indicates if your type is active or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            TaskType tt = ws.task.createType("Type one", true, "Description");
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### updateType

Description:

Method	Return values	Description
<b>updateType(long typeId, boolean active, String name, String description)</b>	<b>TaskType</b>	Update a task type.

 The boolean parameter "active", indicates if your type is active or not.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long typeId = 4; // A valid type id
            TaskType tp = ws.task.getTaskType(typeId);
            System.out.println(tp);
            tp = ws.task.updateType(typeId, false, "Type one", "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**deleteType**

Description:

Method	Return values	Description
<b>deleteType(long typeId)</b>	<b>void</b>	Delete a task type.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long typeId = 4; // A valid type id
            ws.task.deleteType(typeId);
        }
    }
}

```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**getType**

Description:

Method	Return values	Description
<b>getType(long typeId)</b>	<b>TaskType</b>	Return a task type object by id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long typeId = 1; // A valid type id
            TaskType tt = ws.task.getType(typeId);
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getTypes**

Description:

Method	Return values	Description
<b>getTypes(boolean filterActive)</b>	<b>List&lt;TaskType&gt;</b>	Retrieve a list of all the task types.

Example:

```

package com.openkm;

```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (TaskType tt : ws.task.getTypes(true)) {
                System.out.println(tt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**createStatus**

Description:

Method	Return values	Description
<b>createStatus(String name, boolean finish)</b>	<b>TaskStatus</b>	Create a new task status.


Depending your logic, several status can be ending status. For example status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            TaskStatus ts = ws.task.createStatus("cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```


    }
  }
}

```

**updateStatus**

Description:

Method	Return values	Description
<b>updateStatus(long statusId, String name, boolean finish)</b>	<b>TaskStatus</b>	Update a task status.

 Depending your logic, several status can be ending status. For example status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long statusId = 1; // A valid task id
            TaskStatus ts = ws.task.getTaskStatus(statusId);
            System.out.println(ts);
            ts = ws.task.updateStatus(statusId, "cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**deleteStatus**

Description:

Method	Return values	Description
<b>deleteStatus(long statusId)</b>	<b>void</b>	Delete a task status.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long statusId = 4; // A valid status id
            ws.task.deleteStatus(statusId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getStatus**

Description:

Method	Return values	Description
<b>getStatus(long statusId)</b>	<b>TaskStatus</b>	Return a task status object by id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long statusId = 1; // A valid status id
            TaskStatus ts = ws.task.getStatus(statusId);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}

```

### getStatus

Description:

Method	Return values	Description
<b>getStatus()</b>	<b>List&lt;TaskStatus&gt;</b>	Retrieve a list of all the task status.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            for (TaskStatus ts : okmTask.getStatus(null)) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### createNote

Description:

Method	Return values	Description
<b>createNote(long taskId, String text)</b>	<b>TaskNote</b>	Create a new note to a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    long taskId = 1; // A valid task id
    TaskNote tn = ws.task.createNote(taskId, "New note");
    System.out.println(tn);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**updateNote**

Description:

Method	Return values	Description
<b>updateNote(long noteId, String text)</b>	<b>TaskNote</b>	Update note of a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long noteId = 1; // A valid note id
            TaskNote tn = ws.task.updateNote(noteId, "Note updated");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**deleteNote**

Description:

Method	Return values	Description
<b>deleteNote(String token, long noteId)</b>	<b>void</b>	Delete note of a task.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long notelId = 1; // A valid note id
            ws.task.deleteNote(notelId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getNotes

Description:

Method	Return values	Description
<b>getNotes(long taskId)</b>	<b>List&lt;TaskNote&gt;</b>	Retrieve a list of all the notes of a task.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long taskId = 1; // A valid task id
            for (TaskNote tn : ws.task.getNotes(taskId)) {
                System.out.println(tn);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}  
}  
}
```

## UserConfig samples

### Basics



Example of uuid:

- Using UUID -> "373bcdd0-c082-4e7b-addr-e10ef813946e";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "login". You can access the "login" method from webservice object "ws" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the UserConfig methods from "userConfig" class as is shown below:

```
ws.userConfig.getConfig();
```

### Methods

#### getConfig

Description:

Method	Return values	Description
getConfig()	UserConfig	Returns the user settings.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.bean.UserConfig;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            UserConfig userConfig = ws.userConfig.getConfig();
            Node node = ws.node.getNodeByUuid(userConfig.getHomeNode());
            System.out.println("User: " + userConfig.getUser());
            System.out.println("Home: " + node.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**setHome**

Description:

Method	Return values	Description
<b>setHome(String uuid)</b>	<b>void</b>	Set the user home node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.bean.UserConfig;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.userConfig.setHome("53751cb7-c462-4320-ba46-4cc33e4667ae");

            UserConfig userConfig = ws.userConfig.getConfig();
            Node node = ws.node.getNodeByUuid(userConfig.getHomeNode());
            System.out.println("Home: " + node.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Wizard Samples

### Basics



Example of uuid:

- Using UUID -> "373bcdd0-c082-4e7b-addr-e10ef813946e";

### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Wizard methods from "**wizard**" class as is shown below:

```
ws.wizard.findByUser();
```

## Methods

### findByUser

Description:

Method	Return values	Description
<b>findByUser()</b>	<b>List&lt;WizardNode&gt;</b>	Returns a list of nodes with a wizard.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```

import java.util.List;

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);
            List<WizardNode> list = ws.wizard.findByUser();
            for (WizardNode wizardNode : list) {
                System.out.println(wizardNode);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### findAllByUser

Description:

Method	Return values	Description
<b>findAllByUser()</b>	<b>List&lt;WizardNode&gt;</b>	Returns a list of all nodes with an assigned wizard.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

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);
            List<WizardNode> list = ws.wizard.findAllByUser();
            for (WizardNode wizardNode : list) {
                System.out.println(wizardNode);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**findByUserAndUuid**

Description:

Method	Return values	Description
<b>findByUserAndUuid(String uuid)</b>	<b>WizardNode</b>	Get the wizard of a node

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            WizardNode wizardNode = ws.wizard.findByUserAndUuid("11225723-883f-4c12-8816-650b2c82c89b");
            System.out.println(wizardNode);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**hasWizardByUserAndNode**

Description:

Method	Return values	Description
<b>hasWizardByUserAndNode()</b>	<b>boolean</b>	Know if a node has a wizard

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    System.out.println("The node has a wizard: " + ws.wizard.hasWizardByUserAndNode("11225723-883f-4c12-
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**deleteAll**

Description:

Method	Return values	Description
<b>deleteAll()</b>	<b>void</b>	Remove all wizards of the user.

 Only wizards assigned to the user session will be removed.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.deleteAll();
            System.out.println("Delete all");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**postponeNode**

Description:

Method	Return values	Description
--------	---------------	-------------

<b>postponeNode(String uuid, Calendar date)</b>	<b>void</b>	Postpones the node specified UUID to the given date
---	-------------	---

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

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);
            Calendar calendar = Calendar.getInstance();
            calendar.set(Calendar.DAY_OF_MONTH, 15);
            ws.wizard.postponeNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba", calendar);
            System.out.println("postponeNode");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**cancelPostponeNode**

Description:

Method	Return values	Description
<b>cancelPostponeNode(String uuid)</b>	<b>void</b>	Cancels the postponement of the node with the specified UUID

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        ws.wizard.cancelPostponedNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
        System.out.println("cancelPostponedNode");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

**deleteByNode**

Description:

Method	Return values	Description
<b>deleteByNode(String uuid)</b>	<b>void</b>	Delete the wizard node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.deleteByNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
            System.out.println("deleteByNode");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

**addDigitalSignature**

Description:

Method	Return values	Description
<b>addDigitalSignature()</b>	<b>void</b>	Add digital signature in the wizard of a node.

Example:

```

package com.openkm;

```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.addDigitalSignature("055b5206-35d0-4351-ba92-c304bbba7ddf");
            System.out.println("addDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**removeDigitalSignature**

Description:

Method	Return values	Description
<b>removeDigitalSignature</b>	<b>void</b>	Remove digital signature in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeDigitalSignature("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**addShowWizardCategories**

Description:

Method	Return values	Description
<b>addShowWizardCategories</b>	<b>void</b>	Add show wizard categories in the wizard of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeDigitalSignature("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**removeShowWizardCategories**

Description:

Method	Return values	Description
<b>removeShowWizardCategories</b>	<b>void</b>	Remove show wizard categories in the wizard of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeShowWizardCategories("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
        }
    }
}
    
```

```

        System.out.println("removeShowWizardCategories");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**addShowWizardKeywords**

Description:

Method	Return values	Description
<b>addShowWizardKeywords</b>	<b>void</b>	Add show wizard keywords in the wizard of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.addShowWizardKeywords("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("addShowWizardKeywords");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**removeShowWizardKeywords**

Description:

Method	Return values	Description
<b>removeShowWizardKeywords</b>	<b>void</b>	Remove show wizard keywords in the wizard of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;

```

```
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeShowWizardKeywords("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeShowWizardKeywords");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**addShowWizardOCRDataCapture**

Description:

Method	Return values	Description
<b>addShowWizardOCRDataCapture</b>	<b>void</b>	Add show wizard OCR data capture in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.addShowWizardOCRDataCapture("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("addShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

**removeShowWizardOCRDataCapture**

Description:



Method	Return values	Description
<b>removeShowWizardOCRDataCapture</b>	<b>void</b>	Remove show wizard OCR data capture in the wizard of a node.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeShowWizardOCRDataCapture("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**addGroup**

Description:

Method	Return values	Description
<b>addGroup(String uuid, String group)</b>	<b>void</b>	Add metadata group in the wizard.

**i** The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        ws.wizard.addGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl");
        System.out.println("addGroup");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

### removeGroup

Description:

Method	Return values	Description
<b>removeGroup(String uuid, String group)</b>	<b>void</b>	Remove metadata group in the wizard.



The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl");
            System.out.println("removeGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### addWorkflow

Description:

Method	Return values	Description
<b>addWorkflow(String uuid, String workflow)</b>	<b>void</b>	Add a workflow in the wizard.

**i** The parameter **uuid** is the unique node identifier.

The parameter **workflow** is the workflow name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.addWorkflow("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "purchase");
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**removeWorkflow**

Description:

Method	Return values	Description
<b>removeWorkflow(String uuid, String workflow)</b>	<b>void</b>	Remove a workflow in the wizard.

**i** The parameter **uuid** is the unique node identifier.

The parameter **workflow** is the workflow name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.removeWorkflow("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "purchase");
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setAutostart**

Description:

Method	Return values	Description
<b>setAutostart</b>	<b>void</b>	Set auto start to the wizard.



**Autostart should be always set as the last wizard options.**

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ws.wizard.setAutostart("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("setAutostart");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
}  
}
```

## Workflow samples

### Basics

#### Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```



Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Workflow methods from "**workflow**" class as is shown below:

```
ws.workflow.registerProcessDefinition(is);
```

For most examples it has been used the [Purchase workflow sample](#).

### Methods

#### registerProcessDefinition

Description:

Method	Return values	Description
<b>registerProcessDefinition(InputStream is)</b>	<b>void</b>	Registers a new workflow.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```

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);
            InputStream is = new FileInputStream("/home/openkm/Purchase.par");
            ws.workflow.registerProcessDefinition(is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### deleteProcessDefinition

Description:

Method	Return values	Description
<b>deleteProcessDefinition(long pdId)</b>	<b>void</b>	Deletes a workflow.
The parameter pdId value is a valid workflow process definition.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long pdId = 1; // Valid workflow process definition
            ws.workflow.deleteProcessDefinition(pdId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getProcessDefinition

Description:

Method	Return values	Description
<b>getProcessDefinition(long pdId)</b>	<b>void</b>	Returns a workflow process definition.
The parameter pdId value is a valid workflow process definition.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long pdId = 1; // Valid workflow process definition
            ProcessDefinition pd = ws.workflow.getProcessDefinition(pdId);
            System.out.println(pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**runProcessDefinition**

Description:

Method	Return values	Description
<b>runProcessDefinition(long pdId, String uuid, Map&lt;String, String&gt; propertiesMap)</b>	<b>ProcessInstance</b>	Executes a workflow on a node.
The parameter pdId value is a valid workflow process definition.		
The parameter uuid can be any document, mail, folder or record UUID.		
The propertiesMap values are the form element values needed for starting the workflow ( not all workflows need form values for starting ).		

Example:

```

package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long pdId = 8041; // Some valid workflow process definition id

            // Case as part of starting workflow are required form parameter
            Map<String, String> properties = new HashMap<>();
            properties.put("price", "1000");
            properties.put("description", "some description here");

            ws.workflow.runProcessDefinition(pdId, "f86cc22d-9b50-434f-a940-b04cea9c0048", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### findProcessInstances

Description:

Method	Return values	Description
<b>findProcessInstances(long pdId)</b>	<b>List&lt;ProcessInstance&gt;</b>	Retrieves a list of all process instances of some registered workflows definition.
The parameter pdId value is a valid workflow process definition.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        // Get all workflow definitions
        for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
            System.out.println("WF definition: " + pd);

            // Get all process of some workflow definition
            for (ProcessInstance pi : ws.workflow.findProcessInstances(pd.getId())) {
                System.out.println("PI: " + pi);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

**findAllProcessDefinitions**

Description:

Method	Return values	Description
<b>findAllProcessDefinitions()</b>	<b>List&lt;ProcessDefinition&gt;</b>	Retrieves a list of all registered workflows definitions.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
                System.out.println(pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**findLatestProcessDefinitions**

Description:

Method	Return values	Description
<b>findLatestProcessDefinitions()</b>	<b>List&lt;ProcessDefinition&gt;</b>	Retrieves a list of the last workflows definitions.



Several versions of the same workflow can be registered.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Get all latest workflow definitions
            for (ProcessDefinition pd : ws.workflow.findLatestProcessDefinitions()) {
                System.out.println("WF definition: " + pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**findLastProcessDefinition**

Description:

Method	Return values	Description
<b>findLastProcessDefinition(String name)</b>	<b>ProcessDefinition</b>	Retrieves last workflow definition of some specific workflow.

The parameter name identifies an specific workflow definitions group.



Several workflow definition versions that have the same name.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            ProcessDefinition pd = ws.workflow.findLastProcessDefinition("purchase");
            System.out.println("WF definition: " + pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getProcessInstance

Description:

Method	Return values	Description
<b>getProcessInstance(long piId)</b>	<b>ProcessInstance</b>	Returns the process instance.
The parameter piId is a valid process instance id.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long piId = 8108; // Some valid process instance id

```

```

        ProcessInstance pi = ws.workflow.getProcessInstance(pild);
        System.out.println("PI: " + pi);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**findUserTaskInstances**

Description:

Method	Return values	Description
<b>findUserTaskInstances()</b>	<b>TaskInstanceResultSet</b>	Retrieves a list of task instances assigned to the user.

Example:

```

package com.openkm.example;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.bean.workflow.TaskInstanceResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            TaskInstanceResultSet results = ws.workflow.findUserTaskInstances();
            System.out.println("Total: " + results.getTotal());
            for (TaskInstance ti : results.getResults()) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**findTaskInstances**

Description:

Method	Return values	Description
<b>findTaskInstances(long piId)</b>	<b>List&lt;TaskInstance&gt;</b>	Retrieves a list of task instances from a process instance id.

The parameter piId is a valid process instance id.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Get all task instances of some process instance
            long pild = 8108; // Some valid process instance id
            for (TaskInstance ti : ws.workflow.findTaskInstances(pild)) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setTaskInstanceValues**

Description:

Method	Return values	Description
<b>setTaskInstanceValues(long tiId, String transName, Map&lt;String, String&gt; propertiesMap)</b>	<b>void</b>	Set a task instance vales.
<p>The parameter tiId is a valid task instance id.</p> <p>The parameter transName is the choosen transaction.</p> <p>The propertiesMap values are form element values needed for the workflow task ( not all workflow tasks need form values ).</p>		

Example:

```

package com.openkm;

import java.util.HashMap;

```

```

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long pdId = 2;
            long tiId = 8110; // Some valid task instance id
            TaskInstance ti = ws.workflow.getTaskInstance(tiId);
            Map<String, String> properties = new HashMap<>();
            ws.workflow.setTaskInstanceValues(pdId, ti.getId(), ti.getName(), "aprove", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

### getTaskInstance

Description:

Method	Return values	Description
<b>getTaskInstance(long tiId)</b>	<b>TaskInstance</b>	Returns a task instance.
The parameter tiId is a valid task instance id.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long tiId = 8110; // Some valid task instance id
            TaskInstance ti = ws.workflow.getTaskInstance(tiId);
            System.out.println(ti);
        }
    }
}

```

```

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

**startTaskInstance**

Description:

Method	Return values	Description
<b>startTaskInstance(long tiId)</b>	<b>void</b>	Starts a task instance.
The parameter tiId is a valid task instance id.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long tiId = 8110; // Some valid task instance id
            ws.workflow.startTaskInstance(tiId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**setTaskInstanceActorId**

Description:

Method	Return values	Description
<b>setTaskInstanceActorId(long tiId, String actorId)</b>	<b>void</b>	Set the actor to a task instance.
The parameter tiId is a valid task instance id.		

The parameter actorId must be some valid OpenKM userId.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            long tiId = 8110; // Some valid task instance id
            ws.workflow.setTaskInstanceActorId(tiId, "okmAdmin");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**endTaskInstance**

Description:

Method	Return values	Description
<b>endTaskInstance(long tiId, String transName)</b>	<b>void</b>	Ends a task instance.
<p>The parameter tiId is a valid task instance id.</p> <p>The parameter transName is a transaction name.</p>		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
        long tild = 8110; // Some valid task instance id
        ws.workflow.endTaskInstance(tild, "end");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

**getProcessDefinitionForms**

Description:

Method	Return values	Description
<b>getProcessDefinitionForms(long pdId)</b>	<b>Map&lt;String, List&lt;FormElement&gt;&gt;</b>	Return a map with all the process definition forms.
The parameter pdId value is a valid workflow process definition.		

Example:

```

package com.openkm;

import java.util.List;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            Map<String, List<FormElement>> forms = ws.workflow.getProcessDefinitionForms(12);
            for (String key : forms.keySet()) {
                System.out.println("Key:" + key);
                for (FormElement fe : forms.get(key)) {
                    System.out.println("Fe:" + fe);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

**getProcessDefinitionImage**

Description:

Method	Return values	Description
<b>getProcessDefinitionImage(string pdId, string uuid)</b>	<b>String</b>	Returns a workflow diagram in Base64.
The parameter pdId value is a valid workflow process definition.		

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            String tild = "1";
            String res = ws.workflow.getProcessDefinitionImage(tild, "82ed9618-11eb-4b1f-87e7-f853ee60c3c0");
            System.out.println(res);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

**findPooledTaskInstances**

Description:

Method	Return values	Description
<b>findPooledTaskInstances()</b>	<b>TaskInstanceResultSet</b>	Retrieves a list of all pooled task instances.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
    for (TaskInstance ti : ws.workflow.findPooledTaskInstances()) {
        System.out.println(ti);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
```

**findProcessInstancesByNode**

Description:

Method	Return values	Description
<b>findProcessInstancesByNode(String uuid)</b>	<b>List&lt;ProcessInstance&gt;</b>	Retrieves a list of all process instances by node of some registered workflows definition.
The parameter uuid can be any document, mail, folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

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);
            // Get all workflow definitions
            for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
                System.out.println("WF definition: " + pd);

                // Get all process of some workflow definition
                for (ProcessInstance pi : ws.workflow.findProcessInstances(pd.getId())) {
                    System.out.println("PI: " + pi);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
}

```

### getSuggestBoxKeyValue

Description:

Method	Return values	Description
<b>getSuggestBoxKeyValue(long pdId, String uuid, String taskName, String propertyName, String key)</b>	<b>String</b>	Returns the suggestBox value for a key.

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

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);

            long pdId = 4;
            String taskName = "task-elaboration";
            System.out.println(ws.workflow.getSuggestBoxKeyValue(pdId, "301de803-f4ae-48f1-8505-e83b26716956",
            ) catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

### getSuggestBoxKeyValuesFiltered

Description:

Method	Return values	Description
<b>getSuggestBoxKeyValuesFiltered(long pdId, String uuid, String taskName, String propertyName, String filter)</b>	<b>Map&lt;String, String&gt;</b>	Retrieves a map - (key, value) pairs - with suggestBox values

Example:

```


```

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Map;

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);

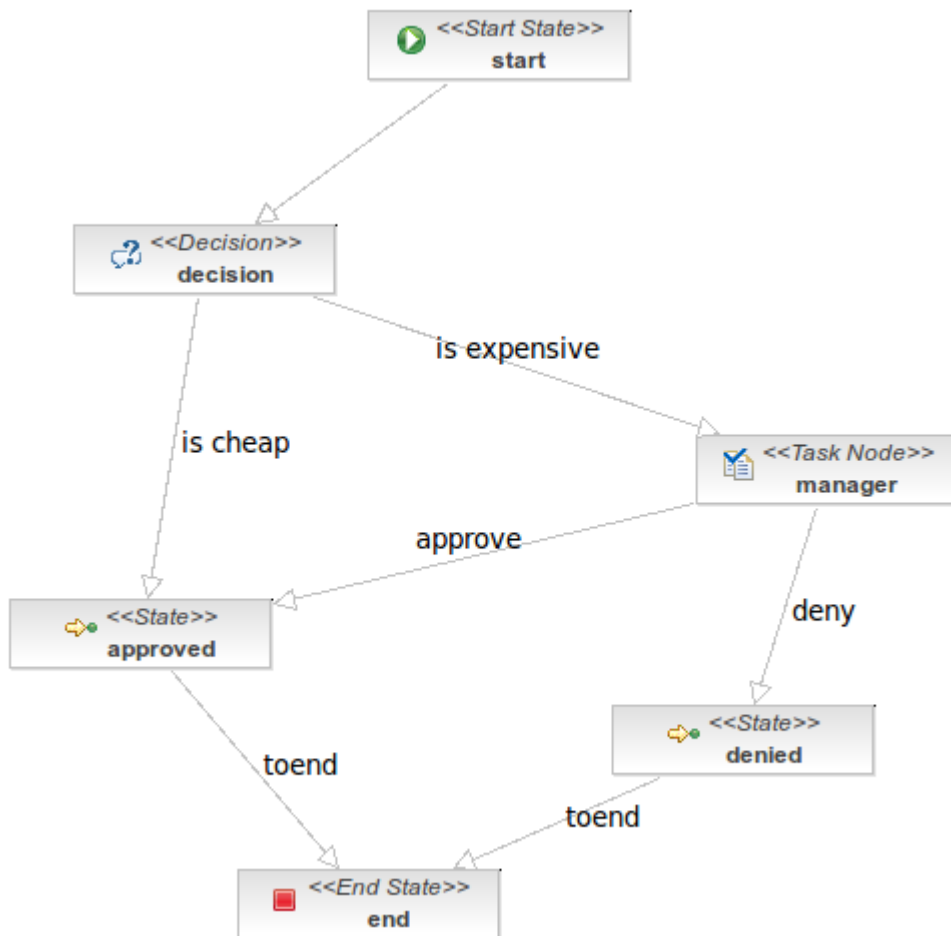
            Map<String, String> values = ws.workflow.getSuggestBoxKeyValuesFiltered(4, "301de803-f4ae-48f1-8505-e
            for (String key : values.keySet()) {
                String value = values.get(key);
                System.out.println(key + ":" + value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Purchase workflow sample

The tasks will be assigned to an user called "manager" so you need to create this user and log as such to see the task assignment. Also you can assign this task to another user from the process instance workflow administration.

Download the [Purchase.par](#) file.

### Process image



### Process definition

```

<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="purchase">
  <start-state name="start">
    <transition to="decision"></transition>
  </start-state>

  <decision name="decision">
    <transition to="approved" name="is cheap">
      <condition expression="#{price.value &lt;= 500}"></condition>
    </transition>
  </decision>

  <task name="manager">
    <assignment user="manager"></assignment>
  </task>

  <state name="approved">
    <transition to="end" name="toend"></transition>
  </state>

  <state name="denied">
    <transition to="end" name="toend"></transition>
  </state>

  <end-state name="end"></end-state>
</process-definition>
  
```

```

    </transition>
    <transition to="manager" name="is expensive">
      <condition expression="#{price.value > 500}"></condition>
    </transition>
  </decision>

  <task-node name="manager">
    <task name="evaluate price">
      <description>The manager may deny purchase or go ahead.</description>
      <assignment actor-id="manager"></assignment>
    </task>
    <transition to="denied" name="deny"></transition>
    <transition to="approved" name="approve"></transition>
  </task-node>

  <state name="approved">
    <description>The purchase has been approved.</description>
    <timer duedate="15 seconds" name="approved timer" transition="toend">
      <script>print(&quot;From APPROVED Go to END&quot;);</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <state name="denied">
    <description>The purchase has been denied.</description>
    <timer duedate="15 seconds" name="denied timer" transition="toend">
      <script>print(&quot;From DENIED Go to END&quot;);</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <end-state name="end"></end-state>
</process-definition>

```

## Process handlers

None.

## Form definition

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
    "http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="run_config">
    <input label="Purchase price" name="price" />
    <textarea label="Purchase description" name="description" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="evaluate price">
    <input label="Purchase price" name="price" data="price" readonly="true" />
    <textarea label="Purchase description" name="description" data="description" readonly="true" />
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>

```