



Documentation for SDK for Java 2.0-ce

Table of Contents

Table of Contents	2
SDK for Java 2.0-ce	10
License	10
Compatibility	10
Sample client	11
Jar sample	12
Basic concepts	14
Authentication	14
Accessing API	15
Class Hierarchy	18
Activity samples	20
Basics	20
Suggested code sample	20
Methods	20
findActivityLog	20
getActivityActions	21
Auth samples	23
Basics	23
Suggested code sample	23
Methods	23
login	24
login	24
logout	25
getGrantedUsersAndRoles	26
getGrantedRoles	27
getGrantedUsers	27
getRoles	28
getRolesByUser	29
getUsers	29
getUser	30
getUsersByRole	31
revokeRole	31
revokeUser	32
grantRole	33
grantUser	34
changeSecurity	34
overwriteSecurity	35
hasSecurityRecursive	36
isAdmin	37
getSessionId	38
createUser	38
deleteUser	39
updateUser	40
createRole	40
deleteRole	41
updateRole	41
assignRole	42
removeRole	42
getProfiles	43
getUserProfile	44
setUserProfile	45
getUserToken	45
setUserPermissions	46
setRolePermissions	47
getUserTenants	48
setUserTenant	48

isLoginLowercase	49
isPasswordExpired	50
getToken	50
Bookmark samples	52
Basics	52
Suggested code sample	52
Methods	52
getUserBookmarks	52
createBookmark	53
renameBookmark	53
deleteBookmark	54
Conversion samples	56
Basics	56
Suggested code sample	56
Methods	56
doc2pdf	56
imageConvert	57
html2pdf	58
doc2txt	59
img2txt	60
barcode2txt	61
Dashboard samples	62
Basics	62
Suggested code sample	62
Methods	62
getUserCheckedOutDocuments	62
getUserLastModifiedDocuments	63
getUserLockedDocuments	64
getUserLockedRecords	65
getUserLockedFolders	67
getUserLockedMails	68
getUserSubscribedDocuments	69
getUserSubscribedFolders	70
getUserSubscribedRecords	71
getUserLastCreatedDocuments	72
getUserLastDocumentsNotesCreated	73
getUserLastCreatedFolders	74
getUserLastFoldersNotesCreated	75
getUserLastCreatedRecords	76
getUserLastRecordsNotesCreated	78
getUserLastDownloadedDocuments	79
getUserLastImportedMails	80
getUserLastMailsNotesCreated	81
getUserLastImportedMailAttachments	82
getUserSearches	83
findUserSearches	84
Document samples	85
Basics	85
Suggested code sample	85
Methods	85
createDocument	85
createDocument	86
deleteDocument	87
getDocumentProperties	88
getContent	88
getContentByVersion	89
getDocumentChildren	90
renameDocument	91
setProperties	92
setLanguage	92
setDocumentTitle	93
checkout	94
cancelCheckout	94

forceCancelCheckout	95
isCheckedOut	96
checkin	97
checkin	97
isLocked	98
getLockInfo	99
purgeDocument	99
moveDocument	100
copyDocument	101
getVersionHistorySize	102
isValidDocument	102
getDocumentPath	103
getDetectedLanguages	104
extendedDocumentCopy	104
getExtractedText	105
setExtractedText	106
getThumbnail	107
createDocumentFromTemplate	108
updateDocumentFromTemplate	109
getCheckedOut	110
createDocument	111
setDocumentDescription	112
createWizardDocument	112
getNumberOfPages	113
getPageAsImage	114
isAttachment	115
getDocumentPdf	115
Folder samples	117
Basics	117
Suggested code sample	117
Methods	117
createFolder	117
getFolderProperties	118
deleteFolder	118
renameFolder	119
moveFolder	120
getFolderChildren	120
isValidFolder	121
getFolderPath	122
copyFolder	122
extendedFolderCopy	123
getContentInfo	124
purgeFolder	125
createMissingFolders	126
setFolderDescription	126
createFolderFromTemplate	127
Import samples	129
Basics	129
Suggested code sample	129
Methods	129
importDocument	129
importFolder	130
Mail samples	132
Basics	132
Suggested code sample	132
Methods	132
getMailProperties	132
deleteMail	133
purgeMail	133
renameMail	134
moveMail	135
copyMail	135
extendedMailCopy	136
getMailChildren	137

isValidMail	138
getMailPath	139
createAttachment	139
deleteAttachment	140
getAttachments	141
sendMailWithAttachments	141
sendMailWithAttachments	142
importEml	143
importMsg	144
setMailTitle	145
sendMail	146
sendMail	146
setMailDescription	147
getMailContent	148
createWizardMail	148
getMailThumbnail	150
getMailAccounts	151
getMailMessages	151
addMailAccount	152
updateMailAccount	153
testMailAccount	154
deleteMailAccount	154
importMailMessages	155
createMailFilter	156
updateMailFilter	156
deleteMailFilter	157
createMailRule	158
updateMailRule	159
deleteMailRule	161
getMailFilterRules	161
getPdf	162
Node samples	164
Basics	164
Suggested code sample	164
Methods	164
getNodeByUuid	164
getVersionHistory	165
restoreVersion	165
renameVersion	166
deleteVersion	167
purgeVersionHistory	167
getChildrenNodesPaginated	168
getChildrenNodesByCategoryPaginated	169
getBreadcrumb	171
subscribe	171
unsubscribe	172
importZip	173
unZip	173
exportZip	174
getNodesFiltered	175
evaluateDownloadZip	175
restore	176
hasNodesLockedByOtherUser	177
lock	177
forceLock	178
unlock	179
forceUnlock	179
isLocked	180
getLockInfo	181
setComment	181
Note samples	183
Basics	183
Suggested code sample	183
Methods	183
addNote	183

getNote	184
deleteNote	185
setNote	185
listNotes	186
getNotesHistory	187
Notification samples	188
Basics	188
Suggested code sample	188
Methods	188
notify	188
Property samples	190
Basics	190
Suggested code sample	190
Methods	190
addCategory	190
removeCategory	191
addKeyword	191
removeKeyword	192
setSigned	193
isSigned	194
PropertyGroup samples	195
Basics	195
Suggested code sample	195
Methods	195
addPropertyGroup	196
removePropertyGroup	197
getPropertyGroups	198
getAllPropertyGroups	199
getPropertyGroupFormDefinition	199
getPropertyGroupFormDefinition	200
getPropertyGroupForm	201
setPropertyGroupProperties	202
hasPropertyGroup	204
getRegisteredPropertyGroupDefinition	204
registerPropertyGroupDefinition	205
getPropertyGroupSuggestions	206
getPropertyGroupProperties	207
getPropertyGroupsByVersion	208
getPropertyGroupPropertiesByVersion	208
getPropertyGroupByVersionForm	209
getPropertyGroup	210
getSuggestBoxKeyValue	210
getSuggestBoxKeyValuesFiltered	211
validateField	213
Record samples	216
Basics	216
Suggested code sample	216
Methods	216
createRecord	216
getRecordProperties	217
deleteRecord	217
purgeRecord	218
renameRecord	219
moveRecord	219
copyRecord	220
isValidRecord	221
getRecordChildren	221
setRecordTitle	222
getRecordPath	223
setRecordDescription	223
extendedRecordCopy	224
createWizardRecord	225
createRecordFromTemplate	226

createMissingRecords	227
Relation samples	229
Basics	229
Suggested code sample	229
Methods	229
getRelationTypes	229
addRelation	230
deleteRelation	231
getRelations	232
getRelationGroups	232
addRelationGroup	233
addNodeToGroup	234
deleteRelationGroup	235
findRelationGroup	235
setRelationGroupName	236
getAllRelationGroups	236
deleteRelationGroupItem	238
Report samples	239
Basics	239
Suggested code sample	239
Methods	239
getReports	239
getReport	240
generateDownloadReportToken	241
executeReport	241
Repository samples	243
Basics	243
Suggested code sample	243
Methods	243
getRootFolder	243
getTrashFolder	244
getTrashFolderBase	244
getTemplatesFolder	245
getPersonalFolder	246
getPersonalFolderBase	246
getMailFolder	247
getMailFolderBase	248
getCategoriesFolder	248
purgeTrash	249
getUpdateMessage	250
getRepositoryUuid	250
hasNode	251
getNodePath	251
getNodeUuid	252
getAppVersion	253
copyAttributes	253
executeScript	254
executeScript	255
executeSqlQuery	256
executeSqlQuery	257
executeHqlQuery	258
executeHqlQuery	259
getTranslations	260
getConfiguration	261
getChangeLog	262
getServerTime()	263
getAvailableLocales	263
Search samples	265
Basics	265
Suggested code sample	268
Methods	269
find	269
find	270

findPaginated	271
findPaginated	272
findSimpleNodeBasePaginated	273
findSimpleNodeBasePaginated	275
getKeywordMap	276
getCategorizedDocuments	278
saveSearch	278
updateSearch	279
getSearch	280
getAllSearchs	281
deleteSearch	281
findByQuery	282
findByQuery	283
findByQueryPaginated	284
findByQueryPaginated	285
findSimpleNodeBaseByQueryPaginated	287
findSimpleNodeBaseByQueryPaginated	288
findWithMetadata	289
findWithMetadata	291
findWithMedataPaginated	292
findWithMedataPaginated	293
getMimeTypes	295
csvExport	296
UserConfig samples	298
Basics	298
Suggested code sample	298
Methods	298
getConfig	298
setHome	299
Wizard Samples	300
Basics	300
Suggested code sample	300
Methods	300
findByUser	300
findByUserAndUuid	301
hasWizardByUserAndNode	301
deleteAll	302
addShowWizardCategories	303
removeShowWizardCategories	303
addShowWizardKeywords	304
removeShowWizardKeywords	305
addGroup	305
removeGroup	306
addWorkflow	307
removeWorkflow	308
setAutostart	308
Workflow samples	310
Basics	310
Suggested code sample	310
Methods	310
registerProcessDefinition	310
deleteProcessDefinition	311
getProcessDefinition	311
runProcessDefinition	312
findProcessInstances	313
findAllProcessDefinitions	314
findLatestProcessDefinitions	315
findLastProcessDefinition	315
getProcessInstance	316
findUserTaskInstances	317
findTaskInstances	317
setTaskInstanceValues	318
getTaskInstance	319
startTaskInstance	320

setTaskInstanceActorId	320
endTaskInstance	321
getProcessDefinitionForms	322
getProcessDefinitionImage	323
findPooledTaskInstances	323
findProcessInstancesByNode	324
getSuggestBoxKeyValue	325
getSuggestBoxKeyValuesFiltered	325
Purchase workflow sample	327
Process image	327
Process definition	327
Process handlers	328
Form definition	328

SDK for Java 2.0-ce

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 2.0-ce should be used:

- **From OpenKM Professional version 7.0.1**

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>2.0-ce</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.getFolderChildren("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>1.8</java.compiler>
    <sdk4j.version>2.0-ce</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
OKMActivity	Manages the activity log.	Yes	ActivityImpl.java	BaseActivity.java
OKMAuth	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
OKMBookmark	Manages the user bookmarks.	Yes	BookmarkImpl.java	BaseBookmark.java
OKMDashboard	Manages all data shown at dashboard.	Yes	DashboardImpl.java	BaseDashboard.java
OKMDocument	Manages documents. For example create, move or delete a document.	Yes	DocumentImpl.java	BaseDocument.java


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

OKMRecord	Manages records. For example create, move or delete a record.	Yes	RecordImpl.java	BaseRecord.java
OKMRelation	Manages relations between nodes. For example create a relation (parent-child) between two documents.	Yes	RelationImpl.java	BaseRelation.java
OKMRepository	A lot of options related with the repository. For example get the properties of main root node (/okm:root).	Yes	RepositoryImpl.java	BaseRepository.java
OKMReport	Manages reports. For example execute reports.	Yes	ReportImpl.java	BaseReport.java
OKMSearch	Manages search feature. For example manage saved queries or perform a new query to the repository.	Yes	SearchImpl.java	BaseSearch.java
OKMUserConfig	Manages user home configuration.	No		
OKMWorkflow	Manages workflows. For example execute a new workflow.	Yes	WorkflowImpl.java	BaseWorkflow.java

Class Hierarchy

Packages detail:

Name	Description
com.openkm	<p>The com.openkm.OKMWebservicesFactory that returns an com.openkm.OKMWebservices object which implements all interfaces.</p> <pre data-bbox="539 573 1401 674">OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);</pre>
com.openkm.sdk4j.bean	<p>Contains all classes result of unmarshalling REST objects.</p>
com.openkm.sdk4j.definition	<p>All interface classes:</p> <ul style="list-style-type: none"> • com.openkm.sdk4j.definition.BaseActivity • com.openkm.sdk4j.definition.BaseAuth • com.openkm.sdk4j.definition.BaseBookmark • com.openkm.sdk4j.definition.BaseConversion • com.openkm.sdk4j.definition.BaseDashboard • com.openkm.sdk4j.definition.BaseDocument • com.openkm.sdk4j.definition.BaseFolder • com.openkm.sdk4j.definition.BaseMail • com.openkm.sdk4j.definition.BaseNode • com.openkm.sdk4j.definition.BaseNote • com.openkm.sdk4j.definition.BaseNotification • com.openkm.sdk4j.definition.BaseProperty • com.openkm.sdk4j.definition.BasePropertyGroup • com.openkm.sdk4j.definition.BaseRecord • com.openkm.sdk4j.definition.BaseRelation • com.openkm.sdk4j.definition.BaseRepository • com.openkm.sdk4j.definition.BaseSearch • com.openkm.sdk4j.definition.BaseWorkflow
com.openkm.sdk4j.impl	<p>All interface implementation classes:</p> <ul style="list-style-type: none"> • com.openkm.sdk4j.impl.ActivityImpl

	<ul style="list-style-type: none"> • com.openkm.sdk4j.impl.AuthImpl • com.openkm.sdk4j.impl.BookmarkImpl • com.openkm.sdk4j.impl.ConversionImpl • com.openkm.sdk4j.impl.DashboardImpl • com.openkm.sdk4j.impl.DocumentImpl • com.openkm.sdk4j.impl.FolderImpl • com.openkm.sdk4j.impl.MailImpl • com.openkm.sdk4j.impl.NodeImpl • com.openkm.sdk4j.impl.NoteImpl • com.openkm.sdk4j.impl.NotificationImpl • com.openkm.sdk4j.impl.PropertyGroupImpl • com.openkm.sdk4j.impl.PropertyImpl • com.openkm.sdk4j.impl.RecordImpl • com.openkm.sdk4j.impl.RelationImpl • com.openkm.sdk4j.impl.RepositoryImpl • com.openkm.sdk4j.impl.SearchImpl • com.openkm.sdk4j.impl.WorkflowImpl
<p>com.openkm.sdk4j.util</p>	<p>A couple of utilities.</p> <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <p> The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields.</p> </div>
<p>com.openkm.sdk4j.exception</p>	<p>All exception classes.</p>

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
findActivityLog(int page, int length, Calendar beginDate, Calendar endDate, String user, String action, String item)	ActivityList	Returns a list of all activity log by date, user, action and item.
The value of the item 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
getActivityActions()	List<String>	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
login(String user, String password)	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
--------	---------------	-------------

login(String user, String password, int expiration, boolean restrictIP)	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 PathNotFoundException /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

logout()	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();
        }
    }
}
    
```

getGrantedUsersAndRoles

Description:

Method	Return values	Description
getGrantedUsersAndRoles(String uuid)	GrantedUsersAndRolesItem	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
getGrantedRoles(String uuid)	Map<String, Integer>	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-addr-e10ef813946e");
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getGrantedUsers

Description:

Method	Return values	Description
getGrantedUsers(String uuid)	Map<String, Integer>	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("373bccdd0-c082-4e7b-addr-e10ef813946e");
            for (String user : grants.keySet()) {
                System.out.println(user + "->" + grants.get(user));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

getRoles

Description:

Method	Return values	Description
getRoles(boolean showAll)	List<String>	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
getRolesByUser(String user)	List<String>	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
getUsers(boolean showAll)	List<CommonUser>	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
getUser(String userId)	CommonUser	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
getUsersByRole(String role)	List<CommonUser>	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
revokeRole(String uuid, String roleId, int permissions, boolean recursive)	void	Remove role grant on a node.

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

When parameter recursive is true, the change will be applied to the node and its 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);
            // 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
revokeUser(String uuid, String user, int permissions, boolean recursive)	void	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
grantRole(String uuid, String role, int permissions, boolean recursive)	void	Add role grant on a node.

The parameter recursive only has sense when the uuid is a folder or a 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 ROLE_USER write grants at the node but not descendants
            ws.auth.grantRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", Permission.ALL_GRANTS, false);
        }
    }
}
```

```

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

```

grantUser

Description:

Method	Return values	Description
grantUser(String uuid, String user, int permissions, boolean recursive)	void	Add user grant on a node.
<p>The parameter recursive only has sense when the uuid is a folder or 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 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
changeSecurity(String uuid, ChangeSecurity changeSecurity)	void	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

overwriteSecurity(String uuid, ChangeSecurity changeSecurity)	void	Overwrite the security of a node.
<div style="border: 1px dashed orange; padding: 10px; background-color: #fff9c4;">  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. </div>		

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);

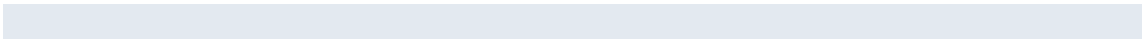
            List<GrantedRole> grList = new ArrayList<>();
            GrantedRole gr = new GrantedRole();
            gr.setRole("ROLE_TEST");
            gr.setPermissions(Permission.READ | Permission.WRITE);
            grList.add(gr);

            ChangeSecurity cs = new ChangeSecurity();
            cs.setGrantedUsersList(guList);
            cs.setGrantedRolesList(grList);
            cs.setRecursive(true);
            ws.auth.overwriteSecurity("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();
        }
    }
}
    
```

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
getSessionId()	String	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
createUser(String userId, String password, String email, String name, boolean active)	void	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
deleteUser(String userId)	void	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
updateUser(String userId, String password, String email, String name, boolean active)	void	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
createRole(String roleId, boolean active)	void	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
deleteRole(String roleId)	void	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
updateRole(String roleId, boolean active)	void	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
assignRole(String userId, String roleId)	void	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
removeRole(String userId, String roleId)	void	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
getProfiles(boolean filterByActive)	List<Profile>	Return the list of all profiles.

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


Each user has assigned one profile that enables more or 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
getUserProfile(String userId)	Profile	Return the profile assigned 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);
            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
setUserProfile(String userId, long profileId)	void	Change the assigned profile to a user.

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

Example:

```

package com.openkm;

import com.openkm.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
getUserToken(String userId)	String	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
setUserPermissions(String uuid, String userId, int permissions, boolean recursive)	void	Update user permissions on a node.
<p>The parameter recursive only has sense when the uuid is a folder or 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);
    // 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
setRolePermissions(String uuid, String roleId, int permissions, boolean recursive)	void	Update role permissions on a node.
<p>The parameter recursive only has sense when the uuid is a folder or 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);
            // 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
getUserTenants()	List<Tenant>	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
setUserTenant(long tenantId)	void	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
isLoginLowercase()	void	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
isPasswordExpired()	void	True when the password of the user has expired.

i 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
getToken()	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();
        }
    }
}
```

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
getUserBookmarks()	List<Bookmark>	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();
    }
}

```

createBookmark

Description:

Method	Return values	Description
createBookmark(String uuid, String name)	void	Create a new bookmark.
The value of the uuid 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.createBookmark("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "test bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

renameBookmark

Description:

Method	Return values	Description
renameBookmark(int bookmarkId, String name)	void	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.renameBookmark(bookmarkId, "set bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteBookmark

Description:

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

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
imageConvert(InputStream is, String fileName, String params, String dstMimeType)	InputStream	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
html2pdf(String url)	InputStream	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
doc2txt(InputStream is, String fileName)	String	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
img2txt(InputStream is, String fileName)	String	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
barcode2txt(InputStream is, String fileName)	String	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
getUserCheckedOutDocuments(int offset, int limit)	DashboardResultSet	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"> • 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. </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
getUserLastModifiedDocuments(int offset, int limit)	DashboardResultSet	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
getUserLockedDocuments(int offset, int	DashboardResultSet	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
getUserLockedRecords(int offset, int limit)	DashboardResultSet	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
getUserLockedFolders(int offset, int limit)	DashboardResultSet	Returns a list of the folders locked by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.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
getUserLockedMails(int offset, int limit)	DashboardResultSet	Returns a list of the mails locked by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```

package com.openkm;

import com.openkm.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
getUserSubscribedDocuments(int offset, int limit)	DashboardResultSet	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
getUserSubscribedFolders(int offset, int limit)	DashboardResultSet	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
getUserSubscribedRecords(int offset, int limit)	DashboardResultSet	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
getUserLastCreatedDocuments(int offset, int limit)	DashboardResultSet	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
getUserLastDocumentsNotesCreated(int offset, int limit)	DashboardResultSet	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.

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.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
getUserLastCreatedFolders(int offset, int limit)	DashboardResultSet	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
--------	---------------	-------------

getUserLastFoldersNotesCreated(int offset, int limit)	DashboardResultSet	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"> • 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. 		
<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"> • limit=10 • offset=0 <p>Now suppose you want to show the results from 11-20, you should use these values:</p> <ul style="list-style-type: none"> • 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.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
getUserLastCreatedRecords(int offset, int limit)	DashboardResultSet	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
getUserLastRecordsNotesCreated(int offset, int limit)	DashboardResultSet	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
getUserLastDownloadedDocuments(int offset, int limit)	DashboardResultSet	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
getUserLastImportedMails(int offset, int limit)	DashboardResultSet	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
getUserLastMailsNotesCreated(int offset, int limit)	DashboardResultSet	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
getUserLastImportedMailAttachments(int offset, int limit)	DashboardResultSet	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
getUserSearches()	List<QueryParams>	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
findUserSearches(long qpId)	List<DashboardNodeResult>	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.createDocument("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is);
```

Methods

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, String name, InputStream is)	Document	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/gnujavasergio/okm/logo.png");
            Document doc = ws.document.createDocument("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is);
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, String name, InputStream is, long nodeClass)	Document	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/gnujavasergio/okm/logo.png");
            long nodeClass = 0;
            Document doc = ws.document.createDocument("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is, n
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteDocument

Description:

Method	Return values	Description
deleteDocument(String uuid)	void	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.deleteDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
        }
    }
}
```

```

        e.printStackTrace();
    }
}

```

getDocumentProperties

Description:

Method	Return values	Description
getDocumentProperties(String uuid)	Document	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.getDocumentProperties("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getContent

Description:

Method	Return values	Description
getContent(String uuid)	InputStream	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
getContentByVersion(String uuid, String versionName)	InputStream	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.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.getContentByVersion("/okm:root/logo.png", "1.1");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentChildren

Description:

Method	Return values	Description
getDocumentChildren(String fldId)	List<Document>	Returns a list of all documents which their parent is fldId.
The parameter fldId can be a folder or a record node 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.getDocumentChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

renameDocument

Description:

Method	Return values	Description
renameDocument(String uuid, String newName)	Document	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.renameDocument("1ec49da9-1746-4875-ae32-9281d7303a62", "logo_renar");
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

setProperties

Description:

Method	Return values	Description
setProperties(String uuid, String title, String description, String lang, List<String> keywords, List<String> categories)	void	Changes some document properties.
<p>The parameter lang must be ISO 691-1 compliant.</p> <div style="border: 1px dashed #ccc; padding: 5px; background-color: #e0f0ff;"> <p>i More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.</p> </div>		

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
--------	---------------	-------------

setLanguage(String uuid, String lang)	void	Sets the document language.
<p>The parameter lang must be ISO 691-1 compliant.</p> <div style="border: 1px dashed blue; padding: 5px; margin: 5px 0;">  More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes. </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.document.setLanguage("1ec49da9-1746-4875-ae32-9281d7303a62", "en");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

setDocumentTitle

Description:

Method	Return values	Description
setDocumentTitle(String uuid, String title)	void	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.setDocumentTitle("1ec49da9-1746-4875-ae32-9281d7303a62", "Some title here");
} catch (Exception e) {
    e.printStackTrace();
}
}
```

checkout

Description:

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

Only one user can modify a document at a time.

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

Example:

```
package com.openkm;

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

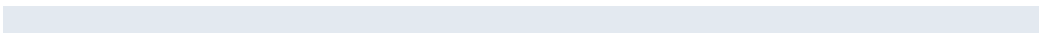
public class Test {

    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
cancelCheckout(String uuid)	void	Cancels a document edition.



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

Example:

```

package com.openkm;

import com.openkm.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
forceCancelCheckout(String uuid)	void	Cancels a document edition.

This method allows to cancel edition on any document.

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



This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```

package com.openkm;

import com.openkm.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
isCheckedOut(String docId)	Boolean	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
checkin(String uuid, InputStream is, String comment)	Version	Updates a document with a new version and returns an object with new Version values.



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

Example:

```

package com.openkm;

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

import org.apache.commons.io.IOUtils;

import com.openkm.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
checkin(String docId, InputStream is, String comment, int increment)	Version	Updates a document with a new version and returns an object with new Version values.

 The 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();
        }
    }
}
```

isLocked

Description:

Method	Return values	Description
isLocked(String uuid)	Boolean	Returns a boolean that indicates if the document is locked 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 document locked:" + ws.document.isLocked("1ec49da9-1746-4875-ae32-9281d7303a62"));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

getLockInfo

Description:

Method	Return values	Description
getLockInfo(String uuid)	LockInfo	Returns an object with the Lock information

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

```

purgeDocument

Description:

Method	Return values	Description
purgeDocument(String uuid)	void	The document is definitely removed from repository.

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



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

Example:

```
package com.openkm;

import com.openkm.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.purgeDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveDocument

Description:

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

Example:

```
package com.openkm;

import com.openkm.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.moveDocument("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab7-ae61-4628-8010-1103");
} catch (Exception e) {
    e.printStackTrace();
}
}
```


copyDocument

Description:

Method	Return values	Description
copyDocument(String uuid, String dstId, String newName)	void	Copy a document into some folder or record.

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

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



Only the binary data and the security grants are copied to destination, the metadata, keywords, etc. of the document are not copied.

See "**extendedDocumentCopy**" 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
getVersionHistorySize(String uuid)	long	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();
        }
    }
}
```

isValidDocument

Description:

Method	Return values	Description
isValidDocument(String docId)	Boolean	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.isValidDocument("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

getDocumentPath

Description:

Method	Return values	Description
getDocumentPath(String uuid)	String	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.getDocumentPath("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

```

    }
  }
}

```

getDetectedLanguages

Description:

Method	Return values	Description
getDetectedLanguages()	List<String>	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();
        }
    }
}

```

extendedDocumentCopy

Description:

Method	Return values	Description
extendedDocumentCopy(String nodeId, String dstId, String name, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)	void	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.



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.extendedDocumentCopy("055b5206-35d0-4351-ba92-c304bbba7ddf",
                System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getExtractedText

Description:

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

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



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

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
setExtractedText(String uuid, InputStream is)	void	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
getThumbnail(String uuid, ThumbnailType type)	InputStream	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();
        }
    }
}
```

createDocumentFromTemplate

Description:

Method	Return values	Description
createDocumentFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map<String, String> properties)	Document	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.

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

- 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:



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.createDocumentFromTemplate("5e72dec8-2409-405b-ac15-d964feb3c");
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

updateDocumentFromTemplate


Description:

Method	Return values	Description
updateDocumentFromTemplate(String uuid, String dstId, Map<String, String> properties)	Document	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.updateDocumentFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "058b9379-b441-4...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

getCheckedOut

Description:

Method	Return values	Description
getCheckedOut()	List<Document>	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();
        }
    }
}

```

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, File file)	Document	Creates a new document and returns as a result 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.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/gnujavasergio/okm/logo.png");
    Document doc = ws.document.createDocument("151f3a54-f370-47d6-801a-d20faecec180", file);
    System.out.println(doc);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

setDocumentDescription

Description:

Method	Return values	Description
setDocumentDescription(String uuid, String description)	void	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.setDocumentDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizardDocument

Description:

Method	Return values	Description
createWizardDocument(String uuid, String name, long nodeClass,	WizardNode	Create a new document

InputStream is)	with wizard.
<p>The parameters uuid should be any valid folder or record UUID.</p> <div style="border: 1px dashed blue; padding: 10px; margin: 10px 0;"> <p>i The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:</p> <ul style="list-style-type: none"> • Add keyword • Add Categories • Add Metadata </div>	

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/gnujavasergio/okm/logo.png");
            WizardNode wn = ws.document.createWizardDocument("1f323e88-64ee-4f57-91e2-9276f8c603f9", "logo.pr
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

getNumberOfPages

Description:

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

Example:

```

package com.openkm;

import com.openkm.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();
        }
    }
}
    
```

isAttachment

Description:

Method	Return values	Description
isAttachment(String docId)	Boolean	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();
        }
    }
}

```

getDocumentPdf

Description:

Method	Return values	Description
getDocumentPdf(String uuid)	InputStream	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.getDocumentProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            InputStream is = ws.document.getDocumentPdf(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();
        }
    }
}
```

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

Methods

createFolder

Description:

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

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

```

getFolderProperties

Description:

Method	Return values	Description
getFolderProperties(String uuid)	Folder	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.getFolderProperties("76f4155e-42af-4be4-9a1c-756497616fb6"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

deleteFolder

Description:

Method	Return values	Description
deleteFolder(String fldId)	void	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.deleteFolder("76f4155e-42af-4be4-9a1c-756497616fb6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

renameFolder

Description:

Method	Return values	Description
renameFolder(String fldId, String newName)	void	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.renameFolder("91264771-02b9-471d-b7a3-ba8cc49a10dd", "renamedFolder");
        // Folder has renamed to /okm:root/renamedFolder
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

moveFolder

Description:

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

Example:

```

package com.openkm;

import com.openkm.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.moveFolder("ada67d44-b081-4b23-bdc1-74181cafb5d", "8599eab7-ae61-4628-8010-1103d6950");
            // Folder has moved to /okm:root/tmp/test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getFolderChildren

Description:

Method	Return values	Description
getFolderChildren(String uuid)	List<Folder>	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.getFolderChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

isValidFolder

Description:

Method	Return values	Description
isValidFolder(String uuid)	Boolean	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.isValidFolder("8cd1e072-8595-4dd3-b121-41d622c43f08");

            // Return true
            ws.folder.isValidFolder("1be884f4-5758-4985-94d1-f18bfe004db8");
        }
    }
}

```

```

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

```

getFolderPath

Description:

Method	Return values	Description
getFolderPath(String uuid)	String	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.getFolderPath("1be884f4-5758-4985-94d1-f18bfe004db8"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


copyFolder

Description:

Method	Return values	Description
copyFolder(String uuid, String dstId, String newName)	void	Copies a folder into a folder or record.

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

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


Only the security grants are copied to destination, the metadata, keywords, etc. of the folder are not copied.

See "**extendedFolderCopy**" 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.copyFolder("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8010-1103d6950c");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

extendedFolderCopy

Description:

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

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

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

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

getContentInfo

Description:

Method	Return values	Description
getContentInfo(String uuid)	ContentInfo	Return and object ContentInfo with information about folder.

The ContentInfo object retrieves information about:

- The number of folders into.
- The number of documents into.
- The number of records into.
- The number of mails into.
- The size in bytes of all objects into the folder.

Example:

```

package com.openkm;

import com.openkm.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();
        }
    }
}


```

purgeFolder

Description:

Method	Return values	Description
purgeFolder(String uuid)	void	The folder is definitely removed from the repository.

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


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

Example:

```

package com.openkm;

import com.openkm.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.purgeFolder("ada67d44-b081-4b23-bdc1-74181cafbc5d");
        }
    }
}

```

```

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

```

createMissingFolders

Description:

Method	Return values	Description
createMissingFolders(String fldPath)	String	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();
        }
    }
}

```

setFolderDescription

Description:

Method	Return values	Description
setFolderDescription(String uuid, String description)	void	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.setFolderDescription("4c195453-246b-4ce9-86ba-b84e68d1f284", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFolderFromTemplate

Description:

Method	Return values	Description
createFolderFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, boolean propertyGroups, boolean security, Map<String, String> properties)	Folder	Creates a new folder from the template and returns an object Folder.

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

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

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.
- When propertyGroups parameter is true the original values of the propertygroups will be copied.
- When security parameter is true the original values of the security 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.createFolderFromTemplate("47c70047-2924-483c-bc42-23d0cbef6b17", "/okm:root");
            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
importDocument(String uuid, File file)	String	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
importFolder(String uuid, String name)	String	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 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 Mail methods from "mail" class as is shown below:

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

Methods

getMailProperties

Description:

Method	Return values	Description
getMailProperties(String uuid)	Mail	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.getMailProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbcd"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteMail

Description:

Method	Return values	Description
deleteMail(String uuid)	void	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.deleteMail("05d9b8e3-f9c1-4ace-9007-afd775dbbcd");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeMail

Description:

Method	Return values	Description
purgeMail(String uuid)	void	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.purgeMail("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameMail

Description:

Method	Return values	Description
renameMail(String uuid, String newName)	void	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.renameMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveMail

Description:

Method	Return values	Description
moveMail(String uuid, String dstId)	void	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.moveMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafbc5d");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyMail

Description:

Return

Method	values	Description
public void copyMail(String uuid, String dstId, String newName)	void	Copies mail into a folder or record.
<p>The values of the dstId parameter should be a folder or record UUID.</p> <p>When parameter newName value is null, mail will preserve the same name.</p> <div style="border: 1px dashed orange; padding: 10px; background-color: #fff9c4;">  Only the security grants are copied to the destination, the metadata, keywords, etc. of the folder are not copied. See "extendedMailCopy" method for this feature. </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.mail.copyMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafb5d", "I
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

extendedMailCopy

Description:

Method	Return values	Description
extendedMailCopy(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security, String newName)	Mail	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.extendedMailCopy("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bc
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailChildren

Description:

Method	Return values	Description
getMailChildren(String uuid)	List<Mail>	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.getMailChildren("ada67d44-b081-4b23-bdc1-74181cafbc5d")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

isValidMail

Description:

Method	Return values	Description
isValidMail(String uuid)	Boolean	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.isValidMail("ada67d44-b081-4b23-bdc1-74181cafbc5d"));

            // Return true
            System.out.println(ws.mail.isValidMail("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
        }
    }
}

```

```

        e.printStackTrace();
    }
}

```

getMailPath

Description:

Method	Return values	Description
getMailPath(String uuid)	String	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.getMailPath("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

createAttachment

Description:

Method	Return values	Description
createAttachment(String uuid, String docName, InputStream is)	Document	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
deleteAttachment(String uuid, String docId)	void	Delete a mail attachment.
The value of the parameter docId parameter can be a valid document 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.deleteAttachment("e3831cc4-30f0-419a-8fbf-a3418472ada5", "6dcb02dc-0881-4610-b2bc-a158ed71");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getAttachments

Description:

Method	Return values	Description
getAttachments(String uuid)	List<Document>	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
sendMailWithAttachments(List<String> to, List<String> cc, List<String> bcc, List<String> replyTo, String subject, String body, List<String> docsId, String uuid)	void	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
sendMailWithAttachments(String from, List<String> to, List<String> cc, List<String> bcc, List<String> replyTo, String subject, String body, List<String> docsId, String uuid)	void	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, "85f07f05
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

importEml

Description:

Method	Return values	Description
importEml(String uuid, String title, InputStream is)	Mail	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
importMsg(String uuid, String title, InputStream is)	Mail	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();
        }
    }
}

```

setMailTitle

Description:

Method	Return values	Description
setMailTitle(String uuid, String title)	void	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.setMailTitle("9d5dd110-db99-4d72-8cf7-610b027a4822", "new title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

sendMail

Description:

Method	Return values	Description
sendMail(List<String> recipients, String subject, String body)	void	Sends a mail.

Example:

```

package com.openkm;

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

import com.openkm.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
sendMail(String from, List<String> recipients, String subject, String body)	void	Sends a mail.
Other parameters:		
<ul style="list-style-type: none"> from is optional and maybe set with empty value 		

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();
        }
    }
}

```

setMailDescription

Description:

Method	Return values	Description
setMailDescription(String uuid, String description)	void	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.setMailDescription("b405a504-d8cb-4166-ac51-22f68acee8c5", "some description");
            Mail mail = ws.mail.getMailProperties("b405a504-d8cb-4166-ac51-22f68acee8c5");
            System.out.println("Description: " + mail.getDescription());
        }
    }
}

```

```

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

```

getMailContent

Description:

Method	Return values	Description
getMailContent(String mailId)	InputStream	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.getMailContent("b405a504-d8cb-4166-ac51-22f68acee8c5");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

createWizardMail

Description:

Method	Return values	Description
createWizardMail(String uuid, String title, InputStream is,	WizardNode	Create a new document with a

String type)		wizard.
<p>The parameters uuid should be any valid folder or record UUID.</p> <p>Available types:</p> <ul style="list-style-type: none"> • Mail.ORIGIN_MSG • Mail.ORIGIN_EML <div style="border: 1px dashed #00aaff; padding: 10px; margin-top: 10px;"> <p>i The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:</p> <ul style="list-style-type: none"> • Add keyword • Add Categories • Add Metadata </div>		

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.createWizardMail("4c195453-246b-4ce9-86ba-b84e68d1f284", "test title", is, Mail.
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```


getMailThumbnail

Description:

Method	Return values	Description
getMailThumbnail(String mailId, ThumbnailType type)	InputStream	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_SEARCH);
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getMailAccounts

Description:

Method	Return values	Description
getMailAccounts()	List<MailAccount>	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.getMailAccounts()) {
                System.out.println(mailAccount);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getMailMessages

Description:

Method	Return values	Description
getMailMessages(long accountId, long start)	MailServerMessages	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();
}
}
```

addMailAccount

Description:

Method	Return values	Description
addMailAccount(MailAccount mailAccount)	void	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.addMailAccount(mailAccount);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

updateMailAccount

Description:

Method	Return values	Description
updateMailAccount(MailAccount mailAccount)	void	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.updateMailAccount(mailAccount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

testMailAccount

Description:

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

testMailAccount(MailAccount mailAccount)	void	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.testMailAccount(mailAccount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

deleteMailAccount

Description:

Method	Return values	Description
deleteMailAccount(long mailAccountId)	void	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.deleteMailAccount(mailAccountId);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

importMailMessages

Description:

Method	Return values	Description
importMailMessages(long mailAccountId, List<Long> messageIds)	void	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.getMailMessages(mailAccountId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                messageIds.add(mail.getUid());
            }

            ws.mail.importMailMessages(mailAccountId, messageIds);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
  }
}

```

createMailFilter

Description:

Method	Return values	Description
createMailFilter(long mailAccountId, MailFilter mailFilter)	void	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.createMailFilter(mailAccountId, filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

updateMailFilter

Description:

Method	Return values	Description
updateMailFilter(MailFilter mailFilter)	void	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.updateMailFilter(filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

deleteMailFilter

Description:

Method	Return values	Description
deleteMailFilter(long mailFilterId)	void	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.deleteMailFilter(filterId);
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```


createMailRule

Description:

Method	Return values	Description
createMailRule(long filterId, MailFilterRule rule)	void	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.createMailRule(filterId, rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

updateMailRule

Description:

Method	Return values	Description
updateMailRule(MailFilterRule rule)	void	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.

i 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.updateMailRule(rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

deleteMailRule

Description:

Method	Return values	Description
deleteMailRule(long ruleId)	void	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.deleteMailRule(ruleId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getMailFilterRules

Description:

Method	Return values	Description
getMailFilterRules(long filterId)	List<MailFilterRule>	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.getMailFilterRules(filterId)) {
        System.out.println(mailFilterRule);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

getPdf

Description:

Method	Return values	Description
getPdf(String uuid)	InputStream	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.getMailProperties("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();
        }
    }
}

```

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>	Node	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
getVersionHistory(String uuid)	List<Version>	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
restoreVersion(String uuid, String versionName)	void	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();
        }
    }
}

```

renameVersion

Description:

Method	Return values	Description
renameVersion(String uuid, String versionName, String newName)	void	Rename 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.renameVersion("34e657ee-0c29-45d8-9c07-c09bbc22076c", "1.3","1.4");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

deleteVersion

Description:

Method	Return values	Description
deleteVersion(String uuid, String versionName)	void	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
purgeVersionHistory(String uuid)	void	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();
        }
    }
}
    
```

getChildrenNodesPaginated

Description:

Method	Return values	Description
getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes)	SimpleNodeBaseList	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
- 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();
        }
    }
}
    
```

getChildrenNodesByCategoryPaginated


Description:

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

<p>getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes)</p>	<p>SimpleNodeBaseList</p>	<p>Get children nodes by category paginated.</p>
--	----------------------------------	--

 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, "", NodesPage
    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
getBreadcrumb(String uuid)	List<BreadcrumbItem>	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

subscribe(String uuid)	void	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
unsubscribe(String uuid)	void	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
importZip(String uuid, InputStream is)	void	Import a zip file.

Example:

```

package com.openkm;

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

import org.apache.commons.io.IOUtils;

import com.openkm.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
unZip(String uuid, String dstId)	void	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
exportZip(List<String> uuids, boolean withPath, boolean background)	InputStream	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();
    }
}
}

```

getNodesFiltered

Description:

Method	Return values	Description
getNodesFiltered(List<String> uuids)	List<Node>	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);

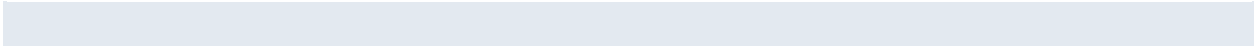
        try {
            ws.login(user, password);
            List<String> uuids = new ArrayList<>();
            uuids.add("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            uuids.add("d386cff8-1d4d-472f-9c6d-f21955ec499a");

            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
evaluateDownloadZip(List<String> uuids)	ZipDownloadEvaluationResult	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();
        }
    }
}
    
```

restore

Description:

Method	Return values	Description
restore(String uuid)	Node	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
hasNodesLockedByOtherUser(String uuid)	boolean	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();
        }
    }
}


```

lock

Description:

Method	Return values	Description

lock(String uuid)	LockInfo	Locks a node and returns an object with the Lock information.
--------------------------	-----------------	---



Only the user who locked the node is allowed to unlock.

A locked node cannot be modified 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);
            ws.node.lock("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

forceLock

Description:

Method	Return values	Description
forceLock(String uuid)	LockInfo	Locks a node and returns an object with the Lock information.

This method allows to lock any node.



This action can only be done by a super user (user with ROLE_ADMIN).

In case exist a previous lock it will be replaced by a new one. When parent is locked you must apply the lock from parent.

Example:

```

package com.openkm;
    
```

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

public class Test {


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

        try {
            ws.login(user, password);
            LockInfo lockInfo = ws.node.forceLock("1ec49da9-1746-4875-ae32-9281d7303a62");
            System.out.println("Author:" + lockInfo.getOwner());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlock

Description:

Method	Return values	Description
unlock(String uuid)	void	Unlocks a locked node.

 Only the user who locked the node is allowed to unlock.

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.unlock("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


forceUnlock

Description:

Method	Return values	Description
forceUnlock(String uuid)	void	Unlocks a locked node.

This method allows to unlock any locked node.

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

 This action can only be done by a 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);
            ws.node.forceUnlock("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

isLocked

Description:

Method	Return values	Description
isLocked(String uuid)	Boolean	Returns a boolean that indicates if the node is locked 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 node locked:" + ws.node.isLocked("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

Method	Return values	Description
getLockInfo(String uuid)	LockInfo	Returns an object with the Lock information

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.getLockInfo("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setComment

Description:



Method	Return values	Description
setComment(String uuid, String versionName, String comment)	void	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();  
        }  
    }  
}
```

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.addNote("373bccdd0-c082-4e7b-addr-e10ef813946e", "the note text");
```

Methods

addNote

Description:

Method	Return values	Description
addNote(String uuid, String text)	Note	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.addNote("373bccdd0-c082-4e7b-addd-e10ef813946e", "the note text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNote

Description:

Method	Return values	Description
getNote(String noteId)	Note	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.listNotes("373bccdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                System.out.println(ws.getNote(notes.get(0).getPath()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
}
}
```

deleteNote

Description:

Method	Return values	Description
deleteNote(String noteId)	Note	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.listNotes("373bccdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.deleteNote(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setNote

Description:

Method	Return values	Description
setNote(String noteId, String text)	void	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.listNotes("373bccdd0-c082-4e7b-addr-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.setNote(notes.get(0).getPath(), "text modified");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

listNotes

Description:

Method	Return values	Description
listNotes(String uuid)	List<Note>	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.listNotes("373bccdd0-c082-4e7b-addr-e10ef813946e");
    for (Note note : notes) {
        System.out.println(note);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

getNotesHistory

Description:

Method	Return values	Description
getNotesHistory(String uuid)	List<NoteHistory>	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.getNotesHistory("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
notify(List<String> uuids, List<String> users, List<String> roles, List<String> mails, String message, boolean attachment)	void	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();
        }
    }
}
```

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
addCategory(String uuid, String catId)	void	Set a relation between a category and a node.
The value of the catId parameter should be a category folder UUID.		

Example:

```

package com.openkm;

import com.openkm.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
removeCategory(String uuid, String catId)	void	Removes a relation between a category and a node.
The value of the catId parameter should be a category folder UUID.		

Example:

```

package com.openkm;

import com.openkm.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
addKeyword(String uuid, String keyword)	void	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
removeKeyword(String uuid, String keyword)	void	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();
        }
    }
}
```

setSigned

Description:

Method	Return values	Description
setSigned(String uuid, boolean signed)	void	Marks a document as signed or unsigned binary data into the repository.
<p>The parameter uuid should be a document node.</p> <div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  <p>This method does not perform any kind of digital signature process, simply mark into the database that a document is signed.</p> </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();
        }
    }
}
```



```

        e.printStackTrace();
    }
}

```

isSigned

Description:

Method	Return values	Description
isSigned(String uuid)	boolean	Returns a boolean that indicates if the document is signed or not.
The parameter uuid should be a document 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);

            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.addPropertyGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", propertiesMap)
```

Methods

addPropertyGroup

Description:

Method	Return values	Description
addPropertyGroup(String uuid, String grpName, Map<String, String> propertiesMap)	void	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.addPropertyGroup("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removePropertyGroup

Description:

Method	Return values	Description
removePropertyGroup(String uuid, String grpName)	void	Removes a metadata group of a node.
The grpName should be a valid Metadata group name.		

Example:

```

package com.openkm;

import com.openkm.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.removePropertyGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getPropertyGroups

Description:

Method	Return values	Description
getPropertyGroups(String uuid)	List<PropertyGroup>	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.getPropertyGroups("8cd1e072-8595-4dd3-b121-41d622c43f")) {
            System.out.println(pGroup);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

getAllPropertyGroups

Description:

Method	Return values	Description
getAllPropertyGroups()	List<PropertyGroup>	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.getAllPropertyGroups()) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

getPropertyGroupFormDefinition

Description:

Method	Return values	Description
getPropertyGroupFormDefinition(String grpName)	List<FormElement>	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

 The method is usually used to display empty form elements for creating new metadata values.

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
getPropertyGroupFormDefinition(String grpName, String uuid)	List<FormElement>	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

 The method is usually used to display empty form elements for creating new metadata values.

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", "8cd1e072-8595-4dd3-b121-41d622"))
                System.out.println(fElement);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm(String uuid, String grpName)	List<FormElement>	Retrieves a list of metadata values 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.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("8cd1e072-8595-4dd3-b121-41d622"))
                System.out.println(fElement);
        }
    }
}

```



```

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


```


setPropertyGroupProperties


Description:

Method	Return values	Description
setPropertyGroupProperties(String uuid, String grpName, Map<String, String> properties)	void	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="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 `"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.setPropertyGroupProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
  }
}

```

hasPropertyGroup

Description:

Method	Return values	Description
hasPropertyGroup(String uuid, String grpName)	Boolean	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.hasPropertyGroup("8cd1e072-8595-4dd3-b
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getRegisteredPropertyGroupDefinition

Description:

Method	Return values	Description
getRegisteredPropertyGroupDefinition()	String	Returns the XML Metada groups definition.


The method only can be executed by super user granted (ROLE_ADMIN member user).

Example:

```

package com.openkm;

import com.openkm.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.getRegisteredPropertyGroupDefinition());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

registerPropertyGroupDefinition

Description:

Method	Return values	Description
registerPropertyGroupDefinition(InputStream is, String name)	void	Set the XML Metada groups definition into the repository.
 The method only can be executed by super user granted (ROLE_ADMIN member user).		

Example:

```

package com.openkm;

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

import 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.registerPropertyGroupDefinition(is, "test");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```


getPropertyGroupSuggestions

Description:

Method	Return values	Description
getPropertyGroupSuggestions(String uuid, String grpName, String propName)	List<String>	Retrieves a list of a suggested metadata field values.

 The propName parameter should be a [Metadata Select field](#) type.

 More information at [Creating your own Suggestion Analyzer](#) and [Metadata Select field](#).

 The sample below is based on this Metadata group definition:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
"http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>

```

Example:

```

package com.openkm;

```

```
import com.openkm.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.getPropertyGroupSuggestions("8cd1e072-8595-4dd3-b121-41d622c43f08"))
                System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupProperties

Description:

Method	Return values	Description
getPropertyGroupProperties(String uuid, String grpName)	Map<String, String>	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.getPropertyGroupProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "p");

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
  }
}

```

getPropertyGroupsByVersion

Description:

Method	Return values	Description
getPropertyGroupsByVersion(String uuid, String versionName)	List<PropertyGroup>	Retrieves a list of metadata groups assigned to a specific version of a node.

Example:

```

package com.openkm;

import com.openkm.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.getPropertyGroupsByVersion("118cb06b-9df7-4d59-bed9-ba
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getPropertyGroupPropertiesByVersion

Description:

Method	Return values	Description
getPropertyGroupPropertiesByVersion(String uuid, String grpName, String versionName)	Map<String, String>	Retrieves a map - (key,value) pairs - with Metadata group values assigned to a specific version of a node.

Example:

```

package com.openkm;

import java.util.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.getPropertyGroupPropertiesByVersion("118cb06b-9df7-4d59-bed9-ba986a4f");

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


```

getPropertyGroupByVersionForm

Description:

Method	Return values	Description
getPropertyGroupByVersionForm(String uuid, String grpName, String versionName)	List<FormElement>	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.getPropertyGroupByVersionForm("118cb06b-9df7-4d59-bed
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getPropertyGroup

Description:

Method	Return values	Description
getPropertyGroup(String grpName)	PropertyGroup	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.getPropertyGroup("okg:consulting"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getSuggestBoxKeyValue

Description:

Return

Method	values	Description
getSuggestBoxKeyValue(String grpName, String propertyName, String key)	String	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:		
<pre><?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></pre>		

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
getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)	Map<String, String>	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_CONSU
      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<String> 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.createRecord("ada67d44-b081-4b23-bdc1-74181cafb5d", "PKI-100200", "new title", 0)
```

Methods

createRecord

Description:

Method	Return values	Description
createRecord(String uuid, String name, String title, long nodeClass)	Record	Creates a new record and return as a result an object Record.
The parameters uuid should be any valid folder or record UUID .		

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.createRecord("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new ti
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getRecordProperties

Description:

Method	Return values	Description
getRecordProperties(String uuid)	Record	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.getRecordProperties("fbc2933e-b141-4557-ab7a-736820ecdb2e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

deleteRecord

Description:

Method	Return values	Description
deleteRecord(String uuid)	void	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.deleteRecord("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

purgeRecord

Description:

Method	Return values	Description
purgeRecord(String uuid)	void	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 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.record.purgeRecord("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameRecord

Description:

Method	Return values	Description
renameRecord(String uuid, String newName)	Record	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.renameRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "new_name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveRecord

Description:

Method	Return values	Description

moveRecord(String uuid, String dstId)	void	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.moveRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d695
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```


copyRecord

Description:

Method	Return values	Description
copyRecord(String uuid, String dstId, String newName)	void	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.copyRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d6950c");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

isValidRecord

Description:

Method	Return values	Description
isValidRecord(String uuid)	Boolean	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.isValidRecord("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

getRecordChildren

Description:

Method	Return values	Description
getRecordChildren(String uuid)	List<Record>	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.getRecordChildren("8599eab7-ae61-4628-8010-1103d6950c63")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRecordTitle

Description:

Method	Return values	Description
setRecordTitle(String uuid, String title)	void	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.setRecordTitle("5489fc37-3eb7-43de-998c-319725ae0ca0", "some title");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

    }
  }
}

```

getRecordPath

Description:

Method	Return values	Description
getRecordPath(String uuid)	String	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.getRecordPath("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

setRecordDescription

Description:

Method	Return values	Description
setRecordDescription(String uuid, String description)	void	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.setRecordDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

extendedRecordCopy

Description:

Method	Return values	Description
extendedRecordCopy(String uuid, String dstId, String newName, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)	Record	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.extendedRecordCopy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7:a
                "new name record", true, true, true, true, true);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

createWizardRecord

Description:

Method	Return values	Description
createWizardRecord(String uuid, String name, String title, long nodeClass)	WizardNode	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.createWizardRecord("1f323e88-64ee-4f57-91e2-9276f8c603f9", "PKI-100200",
    System.out.print(wn);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

createRecordFromTemplate

Description:

Method	Return values	Description
createRecordFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map<String, String> properties)	Record	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.createRecordFromTemplate("9a114b17-7e51-41e7-9d3a-dc7b77e37656", "/okm
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

createMissingRecords

Description:

Method	Return values	Description
createMissingRecords(String recPath)	String	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<RelationType></code>	Retrieves a list of all relations defined of a type.
Available types values: <ul style="list-style-type: none"> • <code>RelationType.BIDIRECTIONAL</code> • <code>RelationType.PARENT_CHILD</code> • <code>RelationType.MANY_TO_MANY</code> 		

 [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();
        }
    }
}
    
```

addRelation

Description:

Method	Return values	Description
addRelation(String nodeAId, String nodeBId, long relTypeId)	void	Sets a relation between two nodes.
The parameters nodeAId and nodeBId should be any valid document, folder, mail, or record UUID .		

Example:

```

package com.openkm;

import com.openkm.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.addRelation("46762f90-82c6-4886-8d21-ad3017dd78a7", "8cd1e072-8595-4dd3-b121-41dc
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

deleteRelation

Description:

Method	Return values	Description
deleteRelation(long relationId)	void	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.deleteRelation(relation.getId());
                }
            }
        } catch (Exception e) {

```

```

        e.printStackTrace();
    }
}
}

```

getRelations

Description:

Method	Return values	Description
getRelations(String uuid)	List<Relation>	Retrieves a list of all relations of a node.
The parameter uuid should be any valid document, folder, mail, or record UUID .		

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
getRelationGroups(String uuid)	List<RelationGroup>	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();
        }
    }
}


```

addRelationGroup

Description:

Method	Return values	Description
addRelationGroup(String uuid, String groupName, long type)	void	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.addRelationGroup("46762f90-82c6-4886-8d21-ad3017dd78a7", "staple group", type.getId());
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}


```

addNodeToGroup

Description:

Method	Return values	Description
addNodeToGroup(String uuid, long groupId)	void	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();
        }
    }
}

```

deleteRelationGroup

Description:

Method	Return values	Description
deleteRelationGroup(long groupId)	void	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.getRelationGroups("8f101a85-88e7-4abe-8175-c5fea3e17d8b")) {
                if (rg.getName().equals("staple group")) {
                    ws.relation.deleteRelationGroup(rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findRelationGroup

Description:

Method	Return values	Description
findRelationGroup(long groupId)	RelationGroup	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.findRelationGroup(groupId));
} catch (Exception e) {
    e.printStackTrace();
}
}
```

setRelationGroupName

Description:

Method	Return values	Description
setRelationGroupName(long groupId, String groupName)	void	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.setRelationGroupName(groupId, "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllRelationGroups

Description:

Method	Return values	Description
getAllRelationGroups(int relationTypeId, String filter, int offset, int limit)	RelationGroupResultSet	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();
        }
    }
}
```

deleteRelationGroupItem

Description:

Method	Return values	Description
deleteRelationGroupItem(String uuid, long groupId)	void	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.deleteRelationGroupItem("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
Date	String	Use the pattern yyyy-MM-dd (year - month - day) 2018-10-04
Select multiple	String	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
getReports(boolean active)	List<Report>	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
getReport(long rpId)	Report	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
generateDownloadReportToken(long rpId)	String	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();
        }
    }
}

```

executeReport

Description:

Method	Return values	Description
executeReport(long rpId, Map<String, String> params, String format, String uuid)	InputStream	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.executeReport(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();
        }
    }
}
```

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
getRootFolder()	Folder	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
getTrashFolder()	Folder	Returns the object Folder of node "/okm:trash/{userId}"

The returned folder will be the user trash folder.


For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:trash/okmAdmin".

Example:

```

package com.openkm;

import com.openkm.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

getTrashFolderBase()	Folder	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
getTemplatesFolder()	Folder	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
getPersonalFolder()	Folder	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.

 For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:personal/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.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
getPersonalFolderBase()	Folder	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
getMailFolder()	Folder	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

 For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:mail/okmAdmin".

Example:

```

package com.openkm;

import com.openkm.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
getMailFolderBase()	Folder	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
getCategoriesFolder()	Folder	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
purgeTrash()	void	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
getUpdateMessage()	String	Retrieves a message when there is a new OpenKM release.

There's an official OpenKM update message service available which is based on your local OpenKM version.

 The most common message is that a new OpenKM version has been released.

Example:

```

package com.openkm;

import com.openkm.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
getRepositoryUuid()	String	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
hasNode(String nodeId)	Boolean	Returns a node that indicate if a node exists or not.
The value of the parameter nodeId can be a valid UUID or path .		

Example:

```
package com.openkm;

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

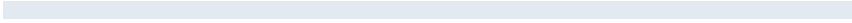
public class Test {

    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("Exists node:" + ws.repository.hasNode("373bcdd0-c082-4e7b-addd-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodePath

Description:



Method	Return values	Description
getNodePath(String uuid)	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("373bcdd0-c082-4e7b-addr-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(String nodePath)	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
getAppVersion()	AppVersion	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
copyAttributes(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes)	void	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
executeScript(InputStream is)	ScriptExecutionResult	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
executeScript(String script)	ScriptExecutionResult	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
executeSqlQuery(InputStream is)	SqlQueryResults	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
executeSqlQuery(String sql)	SqlQueryResults	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
executeHqlQuery(InputStream is)	HqlQueryResults	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
executeHqlQuery(String hql)	HqlQueryResults	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
getTranslations(String lang, String module)	Map<String, String> 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
getConfiguration(String key)	Configuration	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 webservises 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
getChangeLog(String nodePath, Calendar modificationsFrom)	List<ChangeLogged>	Return the list of changes in some path and subfolders.



- The method is used by desktop synchronization application for retrieving the changes.

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.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
getServerTime()	String	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
getAvailableLocales(String locale)	Map<String, String>	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();
        }
    }
}
```


Search samples

Basics


Almosts all methods use QueryParams. Here there're some tips about how using it.

Variables	Type	Allow wildcards	Restrictions
domain	long	No.	<p>Available values:</p> <ul style="list-style-type: none"> • QueryParams.DOCUMENT • QueryParams.FOLDER • QueryParams.MAIL • QueryParams.RECORD <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>
author	String	No.	Value must be a valid userId.
name	String	Yes.	
title	String	Yes.	
keywords	Set<String>	Yes.	
categories	Set<String>	No.	Values should be a category UUID, not use path value.

content		Yes.	
contentType		No.	Value should be a valid and registered MIME type. Only can be applied to documents node.
language		No.	Value should be a valid language. Only can be applied to documents node.
folder		No.	When empty is used by default "/okm:root" node. Value should be a valid UUID, not use a path value.
folderRecursive	Boolean	No.	Only has sense to set this variable to true when the variable folder is not
lastModifiedFrom	Calendar	No.	
lastModifiedTo	Calendar	No.	
mailSubject	String	Yes.	Only apply to mail nodes.

mailFrom	String	Yes.	Only apply to mail nodes.
mailTo		Yes.	Only apply to mail nodes.
notes		Yes.	
properties	Map<String, String>	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<String, String> properties = new HashMap(); properties.put("okp:consulting.name", "name value");</pre> <p>Filtering by range of dates:</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<String,String> properties = new HashMap<>(); 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>To filtering by a metadata field of type multiple like this:</p> <pre><select label="Multiple" name="okp:consulting.multiple" type=" <option label="One" value="one"/> <option label="Two" value="two"/> <option label="Three" value="three" /></pre>

			<pre></select></pre> <p>You should do:</p> <pre>properties.put("okp:consulting.multiple", "one;two");</pre> <p>Where "one" and "two" are valid values and character ";" is used as sep</p>
--	--	--	--

 The search operation is done only by AND logic.

Wildcard examples:

Variable	Example	Description
name	test*.html	Any document that starts with characters "test" and ends with characters ".html"
name	test?.html	Any document that starts with characters "test" followed by a single character and ends with characters ".html"
name	?test*	Any of the documents where the first character doesn't matter, but is followed by the characters, "test".


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
find(QueryParams queryParams, String propertiesPlugin)	List<QueryResult>	Returns a list of results filtered by the values of the queryParams parameter.

 The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

 Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only 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
find(QueryParams queryParams, String sortField, boolean sortReverse, String propertiesPlugin)	List<QueryResult>	Returns a list of results filtered by the values of the queryParams parameter.

 The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

 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();
    }
}
}

```

findPaginated


Description:

Method	Return values	Description
findPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin)	ResultSet	Returns a list of paginated results filtered by the values of the queryParams parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" 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.

 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.impl.OKMWebservices;

public class Test {

    public static void main(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.findPaginated(params, 20, 10, null);
            System.out.println("Total results: " + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated


Description:

Method	Return values	Description
findPaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	ResultSet	Returns a list of paginated results filtered by the values of the queryParams parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" 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.

 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.findPaginated(params, SearchSortField.LAST_MODIFIED, true, 20, 10, null);
            System.out.println("Total results: " + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBasePaginated


Description:

Method	Return values	Description
findSimpleNodeBasePaginated(QueryParams queryParams, int offset, int limit)	SimpleNodeBaseResultSet	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
findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit)	SimpleNodeBaseResultSet	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();
        }
    }
}
```

getKeywordMap

Description:

Method	Return values	Description
getKeywordMap(List<String>	Map<String,	Returns a map of keywords with its count value filtered by

filter)	Integer>	other keywords.
<div style="border: 1px dashed #ccc; padding: 10px; background-color: #e6f2ff;"> <p>i Example:</p> <ul style="list-style-type: none"> • Doc1.txt has keywords "test", "one", "two". • Doc2.txt has keywords "test", "one" • Doc3.txt has keywords "test", "three". <p>The results filtering by "test" -> "one", "two", "three".</p> <p>The results filtering by "one" -> "test", "two".</p> <p>The results filtering by "two" -> "test", "one".</p> <p>The results filtering by "three" -> "test".</p> <p>The results filtering by "one" and "two" -> "test".</p> </div>		

Example:

```

package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
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);
            // All keywords without filtering
            System.out.println("Without filtering");
            Map<String, Integer> keywords = ws.getKeywordMap(new ArrayList<String>());

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used ." + keywords.get(key));
            }

            // Keywords filtered
            System.out.println("Filtering");
            keywords = ws.search.getKeywordMap(Arrays.asList("test"));

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used ." + keywords.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}

```

getCategorizedDocuments

Description:

Method	Return values	Description
getCategorizedDocuments(String categoryId)	List<Document>	Retrieves a list of all documents related with a category.

The values of the categoryId parameter should be a category folder UUID.

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
saveSearch(QueryParams params)	Long	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
updateSearch(QueryParams params)	void	Updates a previously saved search parameters.

 Only can be updated 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);
    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
getSearch(int qpId)	QueryParams	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();
        }
    }
}
```

getAllSearchs

Description:

Method	Return values	Description
getAllSearchs()	List<QueryParams>	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.getAllSearchs()) {
                System.out.println(qParams);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

deleteSearch

Description:

Method	Return values	Description
deleteSearch(int qpId)	void	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 qpld = 1; // Some valid search id
            ws.search.deleteSearch(qpld);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

findByQuery


Description:

Method	Return values	Description
findByQuery(String query, String propertiesPlugin)	List<QueryResult>	Returns a list of results filtered by the query parameter.

 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
findByQuery(String query, String sortField, boolean sortReverse, String propertiesPlugin)	List<QueryResult>	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();
        }
    }
}

```

findByQueryPaginated

Description:

Method	Return values	Description
findByQueryPaginated(String query, int offset, int limit, String propertiesPlugin)	ResultSet	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.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

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

 values:

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```

package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
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);
            ResultSet rs = ws.search.findByQueryPaginated("text:grial AND name:t*.pdf", 0, 10, null);
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

findByQueryPaginated

Description:

Method	Return values	Description
findByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	ResultSet	Returns a list of paginated results filtered by the values of the query parameter.



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](#).



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.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
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);
    ResultSet rs = ws.search.findByQueryPaginated("text:grial AND name:t*.pdf", SearchSortField.NAME, true);
    System.out.println("Total results:" + rs.getTotal());
    for (QueryResult qr : rs.getResults()) {
        System.out.println(qr);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

findSimpleNodeBaseByQueryPaginated

Description:

Method	Return values	Description
findSimpleNodeBaseByQueryPaginated(String query, int offset, int limit)	SimpleNodeBaseResultSet	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:"
            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
findSimpleNodeBaseByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit)	SimpleNodeBaseResultSet	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 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
findWithMetadata(QueryParams queryParams, String propertiesPlugin, List<String> groups)	List<QueryResult>	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
findWithMetadata(QueryParams queryParams, String sortField, boolean sortReverse, String propertiesPlugin, List<String> groups)	List<QueryResult>	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.




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.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
findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List<String> groups)	ResultSet	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
findWithMetadataPaginated(QueryParams queryParams,		Returns a list of paginated results with


int offset, int limit, String propertiesPlugin, List<String> groups)	ResultSet	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, g);
        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
getMimeType()	List<MimeType>	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
csvExport(String lang, QueryParams queryParams, boolean compact)	InputStream	Export as a csv a list of results filtered by the values of the queryParams parameter.
<p>The parameter lang must be ISO 691-1 compliant.</p> <div style="border: 1px dashed blue; padding: 5px; background-color: #e0f0ff;"> <p> More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.</p> </div>		

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();
        }
    }
}

```

```
}  
}
```


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
setHome(String uuid)	void	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
findByUser()	List<WizardNode>	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();
        }
    }
}
```

findByUserAndUuid

Description:

Method	Return values	Description
findByUserAndUuid(String uuid)	WizardNode	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
hasWizardByUserAndNode(String uuid)	boolean	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
deleteAll()	void	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();
}
}
```

addShowWizardCategories

Description:

Method	Return values	Description
addShowWizardCategories(String uuid, int order)	void	Add show wizard categories in the wizard of a node.

i The parameter **order** is the position of the action in the 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);
            ws.wizard.removeDigitalSignature("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", 0);
            System.out.println("removeDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardCategories

Description:

Method	Return values	Description
removeShowWizardCategories(String uuid)	void	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
addShowWizardKeywords(String uuid, int order)	void	Add show wizard keywords in the wizard of a node.

i The parameter **order** is the position of the action in the 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);
            ws.wizard.addShowWizardKeywords("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", 0);
            System.out.println("addShowWizardKeywords");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardKeywords

Description:

Method	Return values	Description
removeShowWizardKeywords(String uuid)	void	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();
        }
    }
}
```

addGroup

Description:

Method	Return values	Description
addGroup(String uuid, String group, int order)	void	Add metadata group in the wizard.

i The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

The parameter **order** is the position of the action in the 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);
            ws.wizard.addGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl", 0);
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

removeGroup

Description:

Method	Return values	Description
removeGroup(String uuid, String group)	void	Remove 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.removeGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl");
            System.out.println("removeGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

addWorkflow

Description:

Method	Return values	Description
addWorkflow(String uuid, String workflow, int order)	void	Add a workflow in the wizard.

i The parameter **uuid** is the unique node identifier.

The parameter **workflow** is the workflow name.

The parameter **order** is the position of the action in the 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);
ws.wizard.addWorkflow("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "purchase", 0);
System.out.println("addGroup");
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

removeWorkflow

Description:

Method	Return values	Description
removeWorkflow(String uuid, String workflow)	void	Remove a workflow in the wizard.

 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
setAutostart	void	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
registerProcessDefinition(InputStream is)	void	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
deleteProcessDefinition(long pdId)	void	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
getProcessDefinition(long pdId)	void	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
runProcessDefinition(long pdId, String uuid, Map<String, String> propertiesMap)	ProcessInstance	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
findProcessInstances(long pdId)	List<ProcessInstance>	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
findAllProcessDefinitions()	List<ProcessDefinition>	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
findLatestProcessDefinitions()	List<ProcessDefinition>	Retrieves a list of the last workflows definitions.

 Several versions of the same workflow can be registered.

Example:

```

package com.openkm;

import com.openkm.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
findLastProcessDefinition(String name)	ProcessDefinition	Retrieves last workflow definition of some specific workflow.

The parameter name identifies an specific workflow definitions group.

 Several workflow definition versions that have the same name.

Example:

```

package com.openkm;

import com.openkm.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
getProcessInstance(long piId)	ProcessInstance	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
findUserTaskInstances()	TaskInstanceResultSet	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
findTaskInstances(long piId)	List<TaskInstance>	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
setTaskInstanceValues(long pdId, long tiId, String tiName, String transName, Map<String, String> propertiesMap)	void	Set a task instance vales.
<p>The parameter pdId is a valid process definition id.</p> <p>The parameter tiId is a valid task instance id.</p> <p>The parameter tiName is the name of the task instance.</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
getTaskInstance(long tiId)	TaskInstance	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 tild = 8110; // Some valid task instance id
        TaskInstance ti = ws.workflow.getTaskInstance(tild);
        System.out.println(ti);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

startTaskInstance

Description:

Method	Return values	Description
startTaskInstance(long tiId)	void	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 tild = 8110; // Some valid task instance id
            ws.workflow.startTaskInstance(tild);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

setTaskInstanceActorId

Description:

Method	Return values	Description
setTaskInstanceActorId(long tiId, String actorId)	void	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
endTaskInstance(long tiId, String transName)	void	Ends a task instance.
<p>The parameter <code>tiId</code> is a valid task instance id.</p> <p>The parameter <code>transName</code> 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
getProcessDefinitionForms(long pdId)	Map<String, List<FormElement>>	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
getProcessDefinitionImage(String pdId, string uuid)	String	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
findPooledTaskInstances()	TaskInstanceResultSet	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
findProcessInstancesByNode(String uuid)	List<ProcessInstance>	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
getSuggestBoxKeyValue(long pdId, String uuid, String taskName, String propertyName, String key)	String	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
getSuggestBoxKeyValuesFiltered(long pdId, String uuid, String taskName, String propertyName, String filter)	Map<String, String>	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);

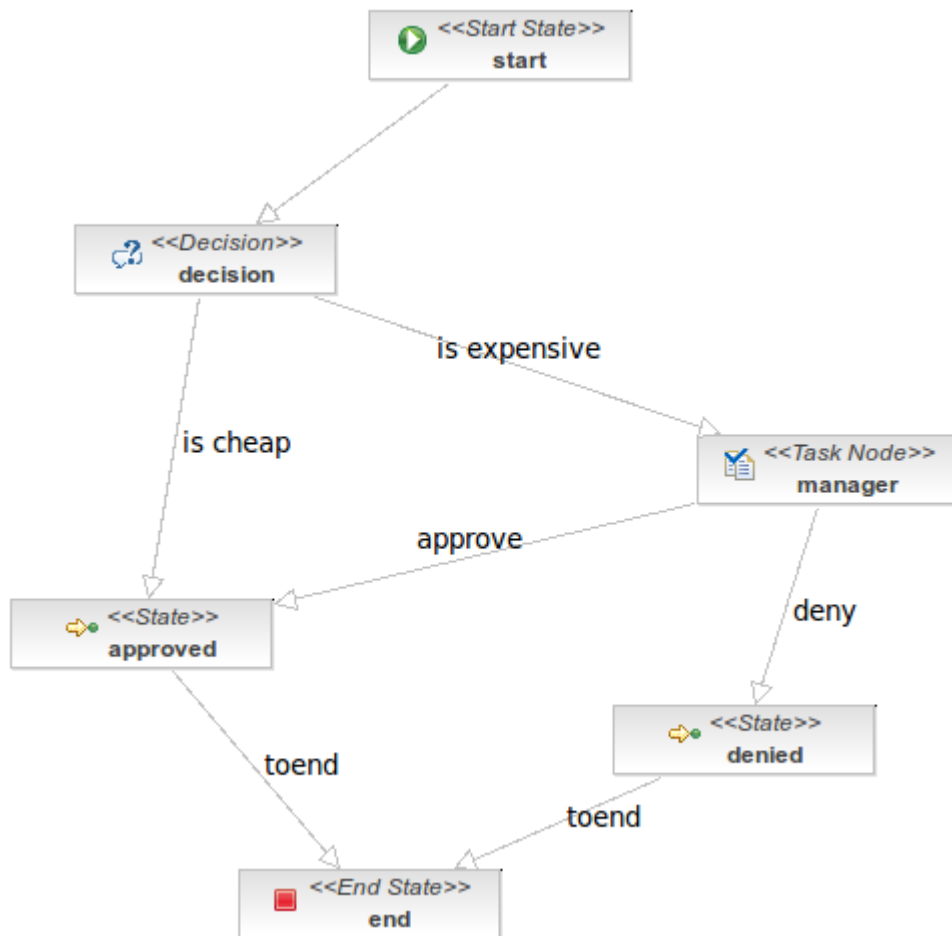
            Map<String, String> values = ws.workflow.getSuggestBoxKeyValuesFiltered(4, "301de803-f4ae-48f1-8505-e
            for (String key : values.keySet()) {
                String value = values.get(key);
                System.out.println(key + ":" + value);
            }
        } catch (Exception e) {
            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">
    <assignee name="manager"></assignee>
  </task>

  <state name="approved">
    <to-end name="toend"></to-end>
  </state>

  <state name="denied">
    <to-end name="toend"></to-end>
  </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>

```