



Documentation for SDK for .NET 4.19.0

Table of Contents

Table of Contents	2
SDK for .NET 4.19.0	12
License	12
Compatibility	12
Download	12
Requirements	13
Visual Studio project configuration	13
Sample client	16
Basic concepts	17
Authentication	17
Accessing API	18
Class Hierarchy	21
Activity samples	24
Basic	24
Suggested code sample	24
Methods	24
findActivityLog	24
getActivityActions	25
Auth samples	27
Basics	27
Suggested code sample	27
Methods	28
login	28
login	29
login	30
logout	31
getRefreshToken	32
getGrantedRoles	32
getGrantedUsers	33
getRoles	34
getRolesByUser	35
getUser	35
getUsers	36
getUsersByRole	37
revokeRole	38
revokeUser	39
grantRole	40
grantUser	40
getProfiles	41
getUserProfile	42
setUserProfile	43
createUser	44
deleteUser	45
updateUser	45
createRole	46
deleteRole	47
updateRole	47
assignRole	48
removeRole	49
changeSecurity	50
overwriteSecurity	50
getGrantedUsersAndRoles	52
setUserPermissions	52
setRolePermissions	53
getUserTenants	54

setUserTenant	55
isAdmin	56
getSessionId	57
hasSecurityRecursive	57
hasTaskManagerAdmin	58
isLoginLowercase	59
isPasswordExpired	60
getUserToken	61
createPersonalAccessToken	61
Bookmark samples	63
Basic	63
Suggested code sample	63
Methods	63
getUserBookmarks	63
create	64
rename	65
delete	65
Conversion samples	67
Basic	67
Suggested code sample	67
Methods	67
doc2pdf	67
imageConvert	68
html2pdf	69
doc2txt	70
img2txt	71
barcode2txt	72
Dashboard samples	74
Basic	74
Suggested code sample	74
Methods	74
getUserCheckedOutDocuments	74
getUserLastModifiedDocuments	75
getUserLockedDocuments	77
getUserLockedRecords	78
getUserLockedFolders	79
getUserLockedMails	80
getUserSubscribedDocuments	81
getUserSubscribedFolders	82
getUserSubscribedRecords	84
getUserLastCreatedDocuments	85
getUserLastDocumentsNotesCreated	86
getUserLastCreatedFolders	87
getUserLastFoldersNotesCreated	89
getUserLastCreatedRecords	90
getUserLastRecordsNotesCreated	91
getUserLastDownloadedDocuments	92
getUserLastImportedMails	93
getUserLastMailsNotesCreated	95
getUserLastImportedMailAttachments	96
getUserSearches	97
findUserSearches	98
Document samples	100
Basics	100
Suggested code sample	100
Methods	100
create	100
create	101
create	102
delete	103
getProperties	104
getContent	104
getContentByVersion	106

getChildren	107
rename	107
setProperties	108
setNodeClass	109
checkout	110
cancelCheckout	111
forceCancelCheckout	111
isCheckedOut	112
checkin	113
checkin	114
purge	115
move	116
copy	116
getVersionHistorySize	117
isValid	118
getPath	119
extendedCopy	119
getExtractedText	121
setExtractedText	122
getThumbnail	122
setLanguage	123
setTitle	124
createFromTemplate	125
updateFromTemplate	126
getDetectedLanguages	127
getAnnotations	128
getDifferences	129
getCheckedOut	130
setDescription	130
setDispositionStage	131
createWizard	132
isOCRDataCaptureSupported	133
recognize	133
captureData	134
getNumberOfPages	135
getPageAsImage	136
mergePdf	136
getLiveEditRestrictedMimeTypes	137
liveEditCheckin	138
liveEditCancelCheckout	139
liveEditForceCancelCheckout	140
liveEditSetContent	140
IsAttachment	141
isConvertibleToPDF	142
getPdf	143
saveAsPdf	144
webPageImport	145
stamp	146
getXrefs	146
refresh	147
FilePlan samples	149
Basic	149
Suggested code sample	149
Methods	149
getRootSections	149
getRootSectionsFilteredBySecurity	150
getChildrenSections	151
getChildrenSectionsFilteredBySecurity	152
getChildrenClasses	153
getChildrenClassesFilteredBySecurity	153
findNodeClassByPk	154
findElectronicRecordClasses	155
findElectronicRecordClassesFilteredBySecurity	156
findFilteredByCodeOrNameFilteredBySecurity	157
findSectionFiltered	158
getNodeClassBreadcrumb	159

findAllNodeClasses	160
getRootSectionsIdWithChildren	160
getChildrenSectionsIdByTenantWithChildren	161
Folder samples	163
Basics	163
Suggested code sample	163
Methods	163
create	163
create	164
getProperties	165
delete	165
rename	166
move	167
getChildren	167
isValid	168
getPath	169
copy	170
extendedCopy	171
getContentInfo	172
purge	173
setStyle	173
createMissingFolders	174
setDescription	175
createFromTemplate	176
Import samples	178
Basic	178
Suggested code sample	178
Methods	178
importDocument	178
importDocument	179
importFolder	180
importFolder	181
importMail	182
importRecord	182
importRecord	183
Mail samples	185
Basics	185
Suggested code sample	185
Methods	185
getProperties	185
delete	186
purge	187
rename	188
move	188
copy	189
extendedCopy	190
getChildren	191
isValid	192
getPath	193
createAttachment	193
deleteAttachment	194
getAttachments	195
sendMailWithAttachments	196
importEml	197
importMsg	198
setTitle	199
sendMail	199
sendMail	200
sendMail	201
setNodeClass	202
setDescription	202
getContent	203
getThumbnail	204
setDispositionStage	205

createWizard	205
getMailsPaginated	206
getAccounts	208
getMailMessages	209
addAccount	209
updateAccount	210
testAccount	211
deleteAccount	212
importMessages	213
createFilter	214
updateFilter	215
deleteFilter	215
createRule	216
updateRule	217
deleteRule	219
getFilterRules	220
forwardEmail	220
getPdf	221
saveAsPdf	222
getExtractedText	223
Node samples	225
Basics	225
Suggested code sample	225
Methods	225
getVersionHistory	225
restoreVersion	226
deleteVersion	227
purgeVersionHistory	227
mayBePromotedAsRecord	228
promoteAsRecord	229
isElectronicRecordPath	230
degradeRecord	230
getElectronicRecordInPath	231
subscribe	232
unsubscribe	232
importZip	233
exportZip	234
unZip	235
getNodeByUuid	235
getChildrenNodesPaginated	236
getChildrenNodesPaginated	237
getChildrenNodesByCategoryPaginated	239
getChildrenNodesByCategoryPaginated	240
getBreadcrumb	242
getNodesFiltered	242
evaluateDownloadZip	243
generateDownloadToken	244
restore	245
hasNodesLockedByOtherUser	245
setComment	246
getPaginatorConfig	247
getPaginatorPlugins	248
lock	249
forceLock	249
unlock	250
forceUnlock	251
Note samples	253
Basics	253
Suggested code sample	253
Methods	253
add	253
get	254
delete	255
set	256
list	257

getNotesHistories	257
Notification samples	259
Basics	259
Suggested code sample	259
Methods	259
notify	259
PDF samples	261
Basics	261
Suggested code sample	261
Methods	261
getImage	261
split	262
extract	263
remove	264
rotate	265
insertPages	266
optimize	268
Plugin samples	269
Basics	269
Suggested code sample	269
Methods	269
executePluginPost	269
executePluginPostReturnFile	270
executePluginGet	272
executePluginGetReturnFile	273
Property samples	277
Basics	277
Suggested code sample	277
Methods	277
addCategory	277
removeCategory	278
addKeyword	279
removeKeyword	280
setEncryption	280
unsetEncryption	281
setSigned	282
isSigned	283
PropertyGroup samples	285
Basics	285
Suggested code sample	285
Methods	286
addGroup	286
removeGroup	287
getGroups	288
getAllGroups	289
getAllGroups	290
getPropertyGroupForm	291
getPropertyGroupFormDefinition	291
getPropertyGroupFormDefinition	292
setProperties	293
hasGroup	295
getRegisteredDefinition	296
registerDefinition	297
getSuggestions	297
getProperties	299
getPropertiesByVersion	300
getGroupsByVersion	300
getGroup	301
getSuggestBoxKeyValue	302
getSuggestBoxKeyValuesFiltered	303
validateField	304

Record samples	307
Suggested code sample	307
Methods	307
create	307
getProperties	308
delete	309
purge	309
rename	310
move	311
copy	312
isValid	313
getChildren	313
setTitle	314
getPath	315
setNodeClass	316
setDispositionStage	316
setDescription	317
createWizard	318
createFromTemplate	319
extendedCopy	320
createMissingRecords	321
Relation samples	323
Basics	323
Suggested code sample	323
Methods	323
getRelationTypes	323
add	324
delete	325
getRelations	326
getRelationGroups	327
addGroup	328
addNodeToGroup	329
deleteGroup	329
getGroup	330
setGroupName	331
getAllRelationGroups	332
deleteItemFromGroup	333
Repository samples	335
Basics	335
Suggested code sample	335
Methods	335
getRootFolder	335
getTrashFolder	336
getTrashFolderBase	337
getTemplatesFolder	337
getPersonalFolder	338
getPersonalFolderBase	339
getMailFolder	340
getMailFolderBase	340
getCategoriesFolder	341
purgeTrash	342
getUpdateMessage	343
getRepositoryUuid	344
hasNode	344
getNodePath	345
getNodeUuid	346
getAppVersion	346
copyAttributes	347
executeScript	348
executeScript	349
executeSqlQuery	350
executeSqlQuery	351
executeHqlQuery	352
executeHqlQuery	353

getTranslations	354
getConfiguration	356
getChangeLog	356
getServerTime()	357
getAvailableLocales	358
getLicenseInfo	359
getClusterUuid	360
getIsFilePlan	360
createShortenUrl	361
Report samples	363
Basics	363
Suggested code sample	363
Methods	363
getReports	363
getReport	364
execute	365
executeSql	367
getSqlReports	367
save	368
saveSql	369
generateDownloadReportToken	370
Stamp samples	372
Basics	372
Suggested code sample	372
Methods	372
getAllStamps	372
testGetStampTextByPk	373
getStampImageByPk	374
getStampBarcodeByPk	374
calculateStampCoordinates	375
calculateStampExpressions	377
stampText	378
stampImage	378
stampBarcode	379
stampImageManually	380
stampTextCustom	381
stampImageCustom	383
calculateFlyImageDimensions	384
getPersonalStamps	385
getPersonalStampImage	386
Search samples	387
Basics	387
Suggested code sample	390
Methods	390
find	390
find	392
findSimpleNodeBasePaginated	393
findSimpleNodeBasePaginated	394
findSimpleNodeBasePaginated	396
getCategorizedDocuments	398
saveSearch	398
updateSearch	399
getSearch	400
getAllSearches	401
deleteSearch	402
getSearchConfig	403
getSearchPlugins	403
findSimpleNodeBaseByQueryPaginated	404
findSimpleNodeBaseByQueryPaginated	406
findByQuery	407
findByQuery	408
findAllDefaultByNodeClass	410
findWithMetadata	410
findWithMetadata	412

findWithMetadataPaginated	413
findWithMetadataPaginated	415
getMimeType	417
csvExport	417
Shard samples	419
Basics	419
Suggested code sample	419
Methods	419
getShards	419
changeShard	421
Task samples	422
Basics	422
Suggested code sample	424
Methods	425
getAssigned	425
getActive	426
getFinished	428
getNotified	430
getStatus	432
getProjects	433
getTypes	433
getAssignedCount	434
getActiveCount	435
getFinishedCount	436
getNotifiedCount	437
create	438
update	440
getTask	442
delete	442
createProject	443
updateProject	444
deleteProject	445
getProject	445
createType	446
updateType	447
deleteType	448
getType	448
createStatus	449
updateStatus	450
deleteStatus	451
getStatus	451
getNotes	452
createNote	453
updateNote	454
deleteNote	454
getHistory	455
getRelatedTasks	456
UserConfig samples	458
Basics	458
Suggested code sample	458
Methods	458
getConfig	458
setHome	459
Wizard samples	461
Basics	461
Suggested code sample	461
Methods	461
findByUser	461
findAllByUser	462
findByUserAndUuid	463
hasWizardByUserAndNode	464
deleteAll	464
postponeNode	465

cancelPostponedNode	466
addDigitalSignature	467
removeDigitalSignature	468
addShowWizardCategories	468
removeShowWizardCategories	469
addShowWizardKeywords	470
removeShowWizardKeywords	471
addShowWizardOCRDataCapture	472
removeShowWizardOCRDataCapture	472
addGroup	473
removeGroup	474
addWorkflow	475
removeWorkflow	476
setAutostart	477
deleteByNode	478
Workflow samples	480
Basics	480
Suggested code sample	480
Methods	480
registerProcessDefinition	480
deleteProcessDefinition	481
getProcessDefinition	482
runProcessDefinition	482
runProcessDefinition	483
findAllProcessDefinitions	484
findProcessInstances	485
findLatestProcessDefinitions	486
findLastProcessDefinition	487
getProcessInstance	488
findUserTaskInstances	489
findTaskInstances	489
setTaskInstanceValues	490
getTaskInstance	491
startTaskInstance	492
setTaskInstanceActorId	493
endTaskInstance	494
getProcessDefinitionForms	494
getProcessDefinitionImage	495
findPooledTaskInstances	496
findProcessInstancesByNode	497
getSuggestBoxKeyValue	498
getSuggestBoxKeyValuesFiltered	499
searchAssignedTasks	499
searchStartedByMe	500
searchOverview	501
Purchase workflow sample	503
Process image	503
Process definition	503
Process handlers	504
Form definition	504

SDK for .NET 4.19.0

OpenKM SDK for .NET is a set of software development tools that allow for the creation of applications for OpenKM. The OpenKM SDK for .NET includes a web services library.

This web services library is a complete API layer for accessing OpenKM through REST web services and provides complete compatibility between OpenKM REST web services versions, minimizing changes to your code.

License



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

This program is distributed WITHOUT ANY WARRANTY; without 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 .NET version 4.19.0 should be used with:

- **OpenKM Professional version 8.1.23 or higher.**

Download

Download professional resources from the OpenKM [Download Center](#).

Requirements

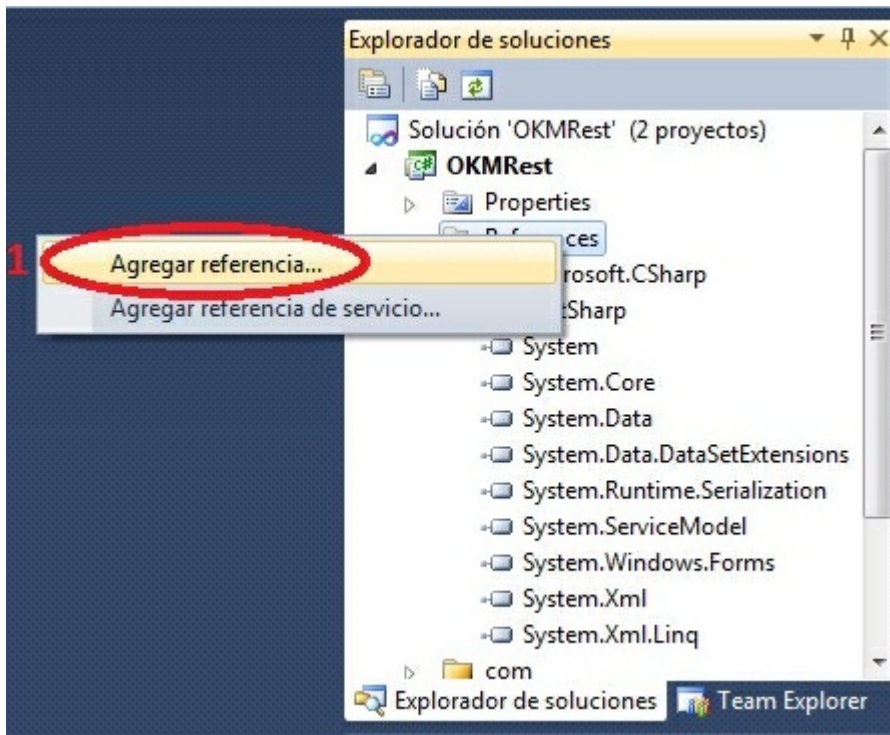
- .NET Framework 4.8.0
- Newtonsoft.Json.dll version 12.0.1
- The .NET SDK requires the RestSharp.dll library version v106.10.1.



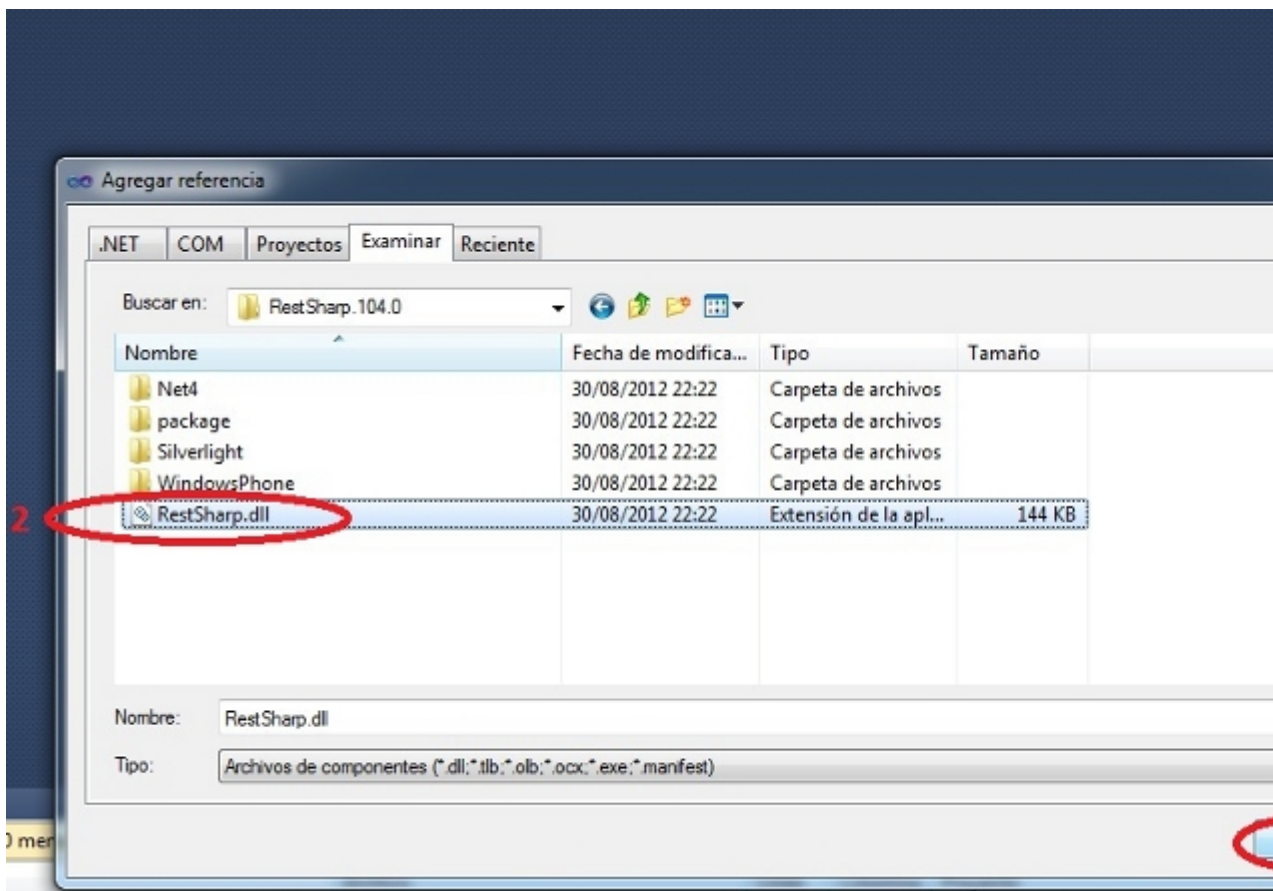
More information about RestSharp is available at <http://restsharp.org/>.

Visual Studio project configuration

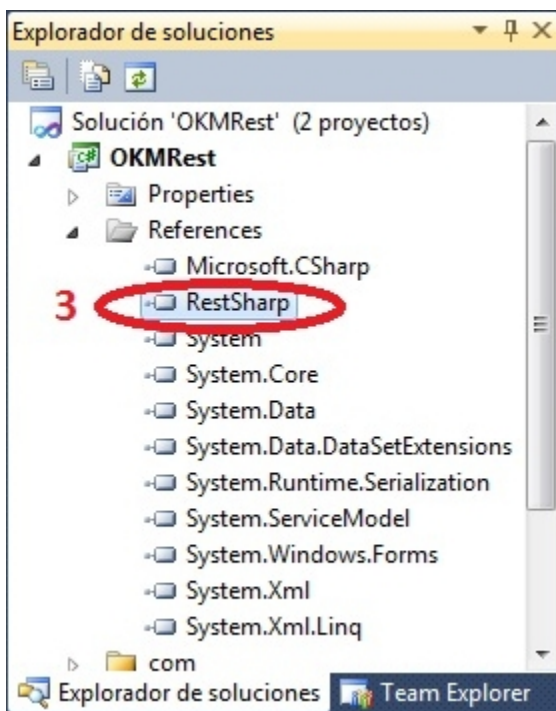
- Download the "**RestSharp.dll**" from [here](#).
- Right-click "**References**" and, in the context menu, choose "**Add Reference**".



- Choose the "**RestSharp.dll**" file from your file system.



- Click the "Accept" button.



The DLL library is now configured in your .NET project.

Sample client

Your first class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    class Program
    {
        /**
         * Sample OpenKM JDK client
         */
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach(Folder fld in ws.folder.getChildren("14530e37-ac51-4a26-8049-0146a5916dec"))
                {
                    System.Console.WriteLine("Folder -> " + fld.path);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }

            System.Console.ReadKey();
        }
    }
}
```

Basic concepts

Authentication

First, you must create the webservice object:

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

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

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



Once you are logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

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

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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.exception;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getAppVersion().getVersion());
            } catch (RepositoryException e) {
                System.Console.WriteLine(e.ToString());
            } catch (DatabaseException e) {
                System.Console.WriteLine(e.ToString());
            } catch (UnknowException e) {
                System.Console.WriteLine(e.ToString());
            } catch (WebserviceException e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

```

        System.Console.ReadKey();
    }
}

```

Accessing API

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



At the main URL <http://docs.openkm.com/javadoc/> you can see all available Javadoc documentation.

At the time of writing this page, the current OpenKM version was 7.1.5, which may change over time.



There is a direct correspondence between the classes and methods in the com.openkm.api packages and those available in the SDK for .NET.

OpenKM API classes:

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

	delete a folder.			
OKMMail	Manages mail. For example, create, move, or delete a mail.	Yes	MailImpl.cs	BaseMail.cs
OKMNode	Manages general-purpose node actions.	Yes	NodeImpl.cs	BaseNode.cs
OKMNote	Manages notes on any node type. For example, create, edit, or delete a note on a document, folder, mail, or record.	Yes	NoteImpl.cs	BaseNote.cs
OKMNotification	Manages notifications. For example, add or remove subscriptions to a document or a folder.	Yes	NotificationImpl.cs	BaseNotification.cs
OKMProperty	Manages categories and keywords. For example, add or remove keywords on a document, folder, mail, or record.	Yes	PropertyImpl.cs	BaseProperty.cs
OKMPropertyGroup	Manages metadata. For example, add a metadata group or set metadata fields.	Yes	PropertyGroupImpl.cs	BasePropertyGroup.cs
OKMRecord	Manages records. For example, create, move, or delete a record.	Yes	RecordImpl.cs	BaseRecord.cs

OKMRelation	Manages relations between nodes. For example, create a relation (parent-child) between two documents.	Yes	RelationImpl.cs	BaseRelation.cs
OKMRepository	A lot of material related to the repository. For example, get the properties of the main root node (/okm:root).	Yes	RepositoryImpl.cs	BaseRepository.cs
OKMReport	Manages reports. For example, execute reports.	Yes	ReportImpl.cs	BaseReport.cs
OKMSearch	Manages the search feature. For example, manage saved queries or perform a new query on the repository.	Yes	SearchImpl.cs	BaseSearch.cs
OKMStamp	Manages stamps.	Yes	StampImpl.cs	BaseStamp.cs
OKMStats	General repository statistics.	No		
OKMTask	Manages tasks. For example, create a new task.	Yes	TaskImpl.cs	BaseTask.cs
OKMUserConfig	Manages the user's home configuration.	No	UserConfigImpl.cs	BaseUserConfig.cs
OKMWorkflow	Manages workflows. For example, execute a new	Yes	WorkflowImpl.cs	BaseWorkflow.cs

workflow.



(*) Indicates that REST support exists for it, but it is still not implemented in the SDK.

Class Hierarchy

Packages detail:

Name	Description
com.openkm.sdk4csharp	The com.openkm.OKMWebservicesFactory that returns a com.openkm.OKMWebservices object which implements all interfaces. <pre>OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);</pre>
com.openkm.sdk4csharp.bean	Contains all classes resulting from unmarshalling REST objects.
com.openkm.sdk4csharp.definition	All interface classes: - com.openkm.sdk4csharp.definition.BaseActivity - com.openkm.sdk4csharp.definition.BaseAuth - com.openkm.sdk4csharp.definition.AutoCAD - com.openkm.sdk4csharp.definition.Bookmark - com.openkm.sdk4csharp.definition.BaseConversion - com.openkm.sdk4csharp.definition.BaseDashboard - com.openkm.sdk4csharp.definition.BaseDocument - com.openkm.sdk4csharp.definition.BaseFolder - com.openkm.sdk4csharp.definition.BaseFilePlan - com.openkm.sdk4csharp.definition.BaseImport - com.openkm.sdk4csharp.definition.BaseMail - com.openkm.sdk4csharp.definition.BaseNode - com.openkm.sdk4csharp.definition.BaseNote - com.openkm.sdk4csharp.definition.BaseNotification - com.openkm.sdk4csharp.definition.BasePdf - com.openkm.sdk4csharp.definition.BasePlugin

	• com.openkm.sdk4csharp.definition.BaseProperty • com.openkm.sdk4csharp.definition.BasePropertyGroup • com.openkm.sdk4csharp.definition.BaseRecord • com.openkm.sdk4csharp.definition.BaseRelation • com.openkm.sdk4csharp.definition.BaseReport • com.openkm.sdk4csharp.definition.BaseRepository • com.openkm.sdk4csharp.definition.BaseSearch • com.openkm.sdk4csharp.definition.BaseShard • com.openkm.sdk4csharp.definition.BaseStamp • com.openkm.sdk4csharp.definition.BaseTask • com.openkm.sdk4csharp.definition.BaseUserConfig • com.openkm.sdk4csharp.definition.BaseWizard • com.openkm.sdk4csharp.definition.BaseWorkflow
com.openkm.sdk4csharp.impl	All interface implementation classes: • com.openkm.sdk4csharp.impl.ActivityImpl • com.openkm.sdk4csharp.impl.AuthImpl • com.openkm.sdk4csharp.impl.AutocadImpl • com.openkm.sdk4csharp.impl.BookmarkImpl • com.openkm.sdk4csharp.impl.ConversionImpl • com.openkm.sdk4csharp.impl.DashboardImpl • com.openkm.sdk4csharp.impl.DocumentImpl • com.openkm.sdk4csharp.impl.FilePlanImpl • com.openkm.sdk4csharp.impl.FolderImpl • com.openkm.sdk4csharp.impl.ImportImpl • com.openkm.sdk4csharp.impl.MailImpl • com.openkm.sdk4csharp.impl.NodeImpl • com.openkm.sdk4csharp.impl.NoteImpl • com.openkm.sdk4csharp.impl.NotificationImpl • com.openkm.sdk4csharp.impl.PdfImpl • com.openkm.sdk4csharp.impl.ShardImpl • com.openkm.sdk4csharp.impl.PropertyGroupImpl • com.openkm.sdk4csharp.impl.PropertyImpl • com.openkm.sdk4csharp.impl.RecordImpl

	• <code>com.openkm.sdk4csharp.impl.RelationImpl</code> • <code>com.openkm.sdk4csharp.impl.RepositoryImpl</code> • <code>com.openkm.sdk4csharp.impl.SearchImpl</code> • <code>com.openkm.sdk4csharp.impl.ShardImpl</code> • <code>com.openkm.sdk4csharp.impl.StampImpl</code> • <code>com.openkm.sdk4csharp.impl.TaskImpl</code> • <code>com.openkm.sdk4csharp.impl.UserConfigImpl</code> • <code>com.openkm.sdk4csharp.impl.WizardImpl</code> • <code>com.openkm.sdk4csharp.impl.WorkflowImpl</code>
<code>com.openkm.sdk4csharp.util</code>	A couple of utilities.  The class <code>com.openkm.sdk4csharp.util.ISO8601</code> should be used to set and parse metadata date fields.
<code>com.openkm.sdk4csharp.exception</code>	All exception classes.

Activity samples

Basic

Suggested code sample

First, you must create the web service object:

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

Then you should log in using the "**login**" method. You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web service, the session is kept in the web service object. Then you can use the other API methods.

At this point, you can use all the Activity methods from the "**activity**" class as 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, DateTime beginDate, DateTime endDate, String user, String action, String item)	ActivityList	Returns a list of all activity logs by date, user, action, and item.
The value of the item is the UUID of the node (document, folder, mail, or record).		

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DateTime beginDate = new DateTime(2019, 7, 1);
                DateTime endDate = new DateTime(2019, 8, 1);
                ActivityList activityList = ws.activity.findActivityLog(0, 10, beginDate, endDate, "testUser", "LOGIN", null);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getActivityActions

Description:

Method	Return values	Description
getActivityActions()	List<String>	Returns a list of all activity log actions.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try

```

```
{
    {
        ws.login(user, password);
        List<string> actions = ws.activity.getActivityActions();
        foreach(string action in actions)
        {
            System.Console.WriteLine(action);
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

Auth samples

Basics

The class **com.openkm.sdk4sharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or retrieve security grants.



To set READ and WRITE access you should do:

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

To check whether you have permission, you should do:

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

In almost all methods, you'll see a parameter named "**nodeId**". The value of this parameter can be a valid node **UUID** (folder, document, mail, record) or a node **path**.



Example of nodeId:

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

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Auth methods of the "**auth**" class, as shown below:

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

Methods

login

Description:

Method	Return values	Description
login(String personalAccessToken)	String	Returns the authorized login token.

 The login token by **default** has an **expiration time of 24 hours**.

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

Each time the login method is executed, the login token is replaced by a new one.

 When a user logs into the application for the first time, an internal method is called that creates user-specific folders, like /okm:trash/userId, etc.

From an API point of view, this method should only be executed the first time a user accesses the API, to create the specific user folder structure.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                String personalAccessToken = "okmpat-ZRjVfMBcCp2tB1T2Dzbq";
                ws.login(personalAccessToken);
            }
        }
    }
}
```

```

        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

login

Description:

Method	Return values	Description
login(String user, String password)	String	Returns the authorized login token.

 The login token by **default** has an **expiration time of 24 hours**.

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

Each time the login method is executed, the login token is replaced by a new one.

 When a user logs into the application for the first time, an internal method is called that creates user-specific folders, like /okm:trash/userId, etc.

From an API point of view, this method should only be executed the first time a user accesses the API, to create the specific user folder structure.

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

login

Description:

Method	Return values	Description
String login(String user, String password, int expiration, bool restrictIp)	String	Login process return authentication token.
 The login token by default has a variable expiration time defined in days. When restrictIP is true, the token will only work from the source IP address. After executing the login method, all subsequent calls will automatically use the authentication token. Each time the login method is executed, the login token is replaced by a new one.		
 When a user logs into the application for the first time, an internal method is called that creates user-specific folders, like /okm:trash/userId, etc. From an API point of view, this method should only be executed the first time a user accesses the API, to create the specific user folder structure. Otherwise you can get errors, for example PathNotExistsException /okm:trash/userId when deleting objects, because the folders have not been created.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";

```

```

String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
int days = 7;
bool restrictIp = true;
try
{
    System.Console.WriteLine("Token: " + ws.login(user, password, days, restrictIp));
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

logout

Description:

Method	Return values	Description
logout()	void	Executes the logout method on the OpenKM side.
 The authentication token will be reset.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.logout();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

getRefreshToken

Description:

Method	Return values	Description
getRefreshToken()	String	Refreshes the current authentication token.



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

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.impl;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
            try  
            {  
                ws.login(user, password);  
                String token = ws.getRefreshToken();  
                System.Console.WriteLine("New token : " + token);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getGrantedRoles

Description:

Method	Return values	Description

getGrantedRoles(String nodeId)	Dictionary<String, int>	Returns the granted roles of a node.
---------------------------------------	--------------------------------------	--------------------------------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
            try
            {
                ws.login(user, password);
                Dictionary<String, int> grants = ws.auth.getGrantedRoles("14530e37-ac51-4a26-8049-0146a5916dec");
                foreach (String role in grants.Keys)
                {
                    System.Console.WriteLine("role ->" + role);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getGrantedUsers

Description:

Method	Return values	Description
getGrantedUsers(String nodeId)	Dictionary<String, int>	Returns the granted users of a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, int> grants = ws.auth.getGrantedUsers("14530e37-ac51-4a26-8049-0146a5916dec");
                foreach (KeyValuePair<string, int> kvp in grants)
                {
                    Console.WriteLine("{0} -> {1}", kvp.Key, kvp.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getRoles

Description:

Method	Return values	Description
getRoles(boolean showAll)	List<String>	Returns the list of all the roles.
When the showAll variable is set to true, it returns all the roles - enabled and disabled - otherwise it only returns the enabled ones.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            foreach (String role in ws.auth.getRoles(false))
            {
                System.Console.WriteLine(role);
            }
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getRolesByUser

Description:

Method	Return values	Description
getRolesByUser(String user)	List<String>	Returns the list of all the roles assigned to a user.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach(String role in ws.auth.getRolesByUser("okmAdmin"))
                {
                    System.Console.WriteLine(role);
                }
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getUser

Description:

Method	Return values	Description
getUser(String userId)	CommonUser	Returns a user.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                CommonUser user = ws.auth.getUser("okmAdmin");
                System.Console.WriteLine(user.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUsers

Description:

Method	Return values	Description
getUsers(boolean showAll)	List<CommonUser>	Returns the list of all the users.
When the showAll variable is set to true, it returns all the users - enabled and disabled - otherwise it only returns the enabled ones.		

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (CommonUser user in ws.auth.getUsers(false))
                {
                    System.Console.WriteLine(user.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getUsersByRole

Description:

Method	Return values	Description
getUsersByRole(String roleId)	List<CommonUser>	Returns the list of all users who have been assigned a role.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        foreach (CommonUser user in ws.auth.getUsersByRole("ROLE_ADMIN"))
        {
            System.Console.WriteLine(user.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

revokeRole

Description:

Method	Return values	Description
revokeRole(String nodeId, String role, int permissions, boolean recursive)	void	Removes a role grant from a node.
The parameter recursive only makes sense when the nodeId is a folder or record node. When the parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Remove ROLE_USER write grants at the node but not descendants
                ws.auth.revokeRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_USER", Permission.ALL_GRA

                // Remove all ROLE_ADMIN grants to the node and descendants
                ws.auth.revokeRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_ADMIN", Permission.ALL_GRA
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

revokeUser

Description:

Method	Return values	Description
revokeUser(String nodeId, String user, int permissions, boolean recursive)	void	Removes a user grant from a node.
The parameter recursive only makes sense when the nodeId is a folder or record node. When the parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Remove john write grants at the node but not descendants
                ws.auth.revokeUser("14530e37-ac51-4a26-8049-0146a5916dec", "john", Permission.ALL_GRANTS, false);

                // Remove all okmAdmin grants at the node and descendants
                ws.auth.revokeUser("14530e37-ac51-4a26-8049-0146a5916dec", "okmAdmin", Permission.ALL_GRANTS, true);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

grantRole

Description:

Method	Return values	Description
grantRole(String nodeId, String role, int permissions, boolean recursive)	void	Adds a role grant to a node.
The parameter recursive only makes sense when the nodeId is a folder or record node. When the parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Add ROLE_USER write grants at the node but not descendants
                ws.auth.grantRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_USER", Permission.ALL_GRAN

                // Add all ROLE_ADMIN grants to the node and descendants
                ws.auth.grantRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_ADMIN", Permission.ALL_GRAN
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

grantUser

Description:

Return

Method	values	Description
grantUser(String nodeId, String userId, int permissions, boolean recursive)	void	Adds a user grant to a node.
The parameter recursive only makes sense when the nodeId is a folder or record node. When the parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Add john write grants at the node but not descendants
                ws.auth.grantUser("14530e37-ac51-4a26-8049-0146a5916dec", "john", Permission.ALL_GRANTS, false)

                // Add all okmAdmin grants at the node and descendants
                ws.auth.grantUser("14530e37-ac51-4a26-8049-0146a5916dec", "okmAdmin", Permission.ALL_GRANTS,
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getProfiles

Description:

Method	Return values	Description
getProfiles(boolean filterByActive)	List<Profile>	Returns the list of all profiles.
When the parameter filterByActive is enabled, the method returns only the active profiles; otherwise it returns all available		

profiles.



Each user is assigned one profile that enables more or fewer OpenKM UI features.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Profile profile in ws.auth.getProfiles(true))
                {
                    System.Console.WriteLine(profile.toString());
                }
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserProfile

Description:

Method	Return values	Description
getUserProfile(String userId)	Profile	Returns the profile assigned to a user.



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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.auth.getUserProfile("okmAdmin").toString());
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setUserProfile

Description:

Method	Return values	Description
setUserProfile(String userId, long profileId)	void	Changes the profile assigned to a user.

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    // Set the profile named "default" to the user
    foreach (Profile profile in ws.auth.getProfiles(true))
    {
        if (profile.name.Equals("Default"))
        {
            ws.setUserProfile("okmAdmin", profile.id);
        }
    }
} catch (Exception e) {
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

createUser

Description:

Method	Return values	Description
createUser(String user, String password, String email, String name, Boolean active)	void	Creates a new user.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.auth.createUser("test", "password.2016", "some@mail.com", "User Name", true);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}  
}
```

deleteUser

Description:

Method	Return values	Description
deleteUser(String userId)	void	Deletes a user.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                ws.auth.deleteUser("test");  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

updateUser

Description:

Method	Return values	Description
updateUser(String userId, String password, String email, String name, Boolean active)	void	Updates a user.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.auth.updateUser("test", "newpassword", "some@mail.com", "Test", false);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createRole

Description:

Method	Return values	Description
createRole(String roleId, boolean active)	void	Creates a new role.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
        try
        {
            ws.login(user, password);
            ws.auth.createRole("ROLE_TEST", true);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

deleteRole

Description:

Method	Return values	Description
deleteRole(String roleId)	void	Deletes a role.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.auth.deleteRole("ROLE_TEST");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

updateRole

Description:

Method	Return values	Description
updateRole(String roleId, boolean active)	void	Updates a role.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.auth.updateRole("ROLE_TEST", true);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

assignRole

Description:

Method	Return values	Description
assignRole(String userId, String roleId)	void	Assigns a role to a user.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
```

```

static void Main(string[] args)
{
    String host = "http://localhost:8080/openkm";
    String username = "okmAdmin";
    String password = "admin";
    OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

    try
    {
        ws.login(user, password);
        ws.auth.assignRole("test", "ROLE_USER");
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

removeRole

Description:

Method	Return values	Description
removeRole(String userId, String roleId)	void	Removes a role from a user.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.auth.removeRole("test", "ROLE_USER");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

changeSecurity

Description:

Method	Return values	Description
changeSecurity(ChangeSecurity changeSecurity)	void	Changes the security of a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Folder fld = ws.folder.createFolder("a978f8e6-aa87-4f0f-b7bc-6f44cb090fdf", "Folder1");
                ChangeSecurity cs = new ChangeSecurity();
                List<GrantedRole> grList = new List<GrantedRole>();
                GrantedRole gr = new GrantedRole();
                gr.role = "ROLE_TEST";
                gr.permissions = Permission.READ | Permission.WRITE;
                grList.Add(gr);
                cs.grantedRolesList = grList;
                ws.auth.changeSecurity(fld.uuid, cs);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

overwriteSecurity

Description:

Method	Return values	Description
overwriteSecurity(String uuid, ChangeSecurity	void	Overwrites the security of a

changeSecurity)

node.



The ChangeSecurity object is used in the changeSecurity and overwriteSecurity methods.

Although values set in the revokeUsers and revokeRoles variables of the ChangeSecurity object are present, these values will not be taken into consideration when overwriting the security.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                GrantedUser gu = new GrantedUser();
                gu.user = pherrera;
                gu.permissions = Permission.READ | Permission.WRITE;

                List<GrantedUser> guList = new List<GrantedUser>();
                guList.Add(gu);

                GrantedRole gr = new GrantedRole();
                gr.role = "ROLE_TEST";
                gr.permissions = Permission.READ | Permission.WRITE;

                List<GrantedRole> grList = new List<GrantedRole>();
                grList.Add(gr);

                ChangeSecurity changeSecurity = new ChangeSecurity();
                changeSecurity.recursive = false;
                changeSecurity.grantedUsersList = guList;
                changeSecurity.grantedRolesList = grList;

                ws.auth.overwriteSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", changeSecurity);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getGrantedUsersAndRoles

Description:

Method	Return values	Description
getGrantedUsersAndRoles(String uuid)	GrantedUsersAndRolesItem	Returns the granted users and roles of a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                GrantedUsersAndRolesItem grants = ws.auth.getGrantedUsersAndRoles("15e35aef-f1bb-451e-ac86-3b");
                foreach (KeyValuePair<string, int> user in grants.grantedUsers)
                {
                    System.Console.WriteLine(user.Key + "->" + user.Value);
                }
                foreach (KeyValuePair<string, int> role in grants.grantedRoles)
                {
                    System.Console.WriteLine(role.Key + "->" + role.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setUserPermissions

Description:

Method	Return values	Description

setUserPermissions(String uuid, String userId, int permissions, bool recursive)	void	Updates user permissions on a node.
The parameter recursive only makes sense when the uuid is a folder or record node. When the parameter recursive is true, the change will be applied to the node and its descendants.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Set user permissions
                ws.auth.setUserPermissions("22c1d190-f798-489d-b420-2008cb38705b", "user1", Permission.READ + P

                // Update permissions of okmAdmin at the node and descendants
                ws.auth.setUserPermissions("22c1d190-f798-489d-b420-2008cb38705b", "okmAdmin", Permission.ALL_
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setRolePermissions

Description:

Method	Return values	Description
setRolePermissions(String uuid, String roleId, int permissions, bool recursive)	void	Updates role permissions on a node.

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Set role permissions
                ws.auth.setRolePermissions("22c1d190-f798-489d-b420-2008cb38705b", "ROLE_USER", Permission.RI

                // Update permissions of ROLE_ADMIN at the node and descendants
                ws.auth.setRolePermissions("22c1d190-f798-489d-b420-2008cb38705b", "ROLE_ADMIN", Permission.A
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserTenants

Description:

Method	Return values	Description
getUserTenants()	List<Tenant>	Returns the list of all tenants.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Tenant> tenants = ws.auth.getUserTenants();
                foreach (Tenant tenant in tenants)
                {
                    System.Console.WriteLine(tenant.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setUserTenant

Description:

Method	Return values	Description
setUserTenant(long tenantId)	void	Change the assigned tenant for a user.
 A user might have access to several tenants but have only one assigned.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            long tenantId = 1; // Valid tenant id
            ws.auth.setUserTenant(tenantId);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

isAdmin

Description:

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                bool isAdmin = ws.auth.isAdmin();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getSessionId

Description:

Method	Return values	Description
getSessionId()	String	Get the HTTP session ID.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String id = ws.auth.getSessionId();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

hasSecurityRecursive

Description:

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

The role is set in the configuration parameter named "**default.security.recursive.role.**"

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                bool hasSecurityRecursive = ws.auth.hasSecurityRecursive();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

hasTaskManagerAdmin

Description:

Method	Return values	Description
hasTaskManagerAdmin()	bool	Check if the user is a member of the "task manager admin" role.
 The configuration parameter named " default.task.manager.admin.role " is used to indicate the role.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                bool hasSecurityRecursive = ws.auth.hasTaskManagerAdmin();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

isLoginLowercase

Description:

Method	Return values	Description
isLoginLowercase()	bool	Return true when the user's login must be in lowercase.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.auth.isLoginLowercase().ToString());
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```

        System.Console.WriteLine(e.ToString());
    }
}
}

```

isPasswordExpired

Description:

Method	Return values	Description
isPasswordExpired()	Boolean	True when the user's password has expired.



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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Password expired= " + ws.auth.isPasswordExpired().ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getUserToken

Description:

Method	Return values	Description
getUserToken(String userId)	String	Return the auth token for a given user.



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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String userToken = ws.auth.getUserToken("testUser");
                Console.WriteLine("UserToken: " + userToken);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createPersonalAccessToken

Description:

Method	Return values	Description
createPersonalAccessToken(String name, DateTime expiration)	String	Return the personal access token.



You can generate a personal access token for each application that requires access to the API.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

            try
            {
                DateTime today = DateTime.Now;
                DateTime futureDate = today.AddDays(365); // create an expired token
                String personalAccessToken = ws.auth.createPersonalAccessToken("good-pat-test", futureDate);
                OKMWebservices ws = OKMWebservicesFactory.newInstance(Config.HOST);
                ws.login(personalAccessToken);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Bookmark samples

Basic

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

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

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

Methods

getUserBookmarks

Description:

Method	Return values	Description
getUserBookmarks()	List<Bookmark>	Returns a list of all the bookmarks for a user.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
```

```

    {
        String host = "http://localhost:8180/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            List<Bookmark> bookmarks = ws.bookmark.getUserBookmarks();
            foreach (Bookmark bm in bookmarks)
            {
                System.Console.WriteLine(bm.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

create

Description:

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Bookmark bookmark = ws.bookmark.create("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "Any name");
            }
        }
    }
}

```

```
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(int bookmarkId, String name)	void	Renames a bookmark.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int bookmarkId = 1;
                ws.bookmark.rename(bookmarkId, "New name");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

delete

Description:

Method	Return values	Description

delete(int bookmarkId)	void	Deletes a bookmark.
-------------------------------	-------------	---------------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int bookmarkId = 1;
                ws.bookmark.delete(bookmarkId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Conversion samples

Basic

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in to the web service, the session is kept in the web service object. Then you can use the other API methods.

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

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

Methods

doc2pdf

Description:

Method	Return values	Description
doc2pdf(FileStream is, String fileName)	Stream	Retrieves the uploaded document converted to PDF format.
The parameter fileName is the document file name. The application uses this parameter to identify the document MIME type by the document extension.		
 The OpenOffice service must be enabled on the OpenKM server for it to run.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream inputFile = new FileStream("C:\\Documents\\test.docx", FileMode.Open);
                Stream tmpFile= ws.conversion.doc2pdf(inputFile, "test.docx");
                inputFile.Close();
                FileStream outputFile = new FileStream("C:\\Documents\\out.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
                tmpFile.CopyTo(outputFile);
                outputFile.Close();
                tmpFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

imageConvert

Description:

Method	Return values	Description
imageConvert(FileStream fs, String fileName, List<String> params, String dstMimeType)	Stream	Retrieve the uploaded image with transformations.
The variable fileName is the document file name. The application uses this variable to identify the document MIME type by the document extension. The parameter dstMimeType is the expected document MIME type of the result.		
 Using this method, you are actually executing the ImageMagick convert tool on the server side. You can set many parameters - transformations - in the params variable. Take a look at ImageMagick		

[convert tool](#) to get a complete list of them.



The image convert tool must be enabled on the OpenKM server for it to run.

When the params value is not empty, it must always end with the chain "\${fileIn} \${fileOut}".

Ensure there is only one whitespace character as a separator between two parameters.

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

Example:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("C:\\Documents\\test.jpg", FileMode.Open, FileAccess.Read);
                List<string> param = new List<string>();
                param.Add("-resize 50% ${fileIn} ${fileOut}");
                Stream tmpFile = ws.conversion.imageConvert(fileStream, test.jpg, param, "image/png");
                fileStream.Close();
                FileStream destFile = new FileStream("C:\\Documents\\test.png", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

html2pdf

Description:

Method	Return values	Description

html2pdf(String url)	Stream	Retrieves the PDF from a URL.
 The HTML to PDF conversion tool must be enabled on the OpenKM server for it to run.		

Example:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream tmpFile = ws.conversion.html2pdf("http://www.openkm.com");
                FileStream destFile = new FileStream("C:\\Documents\\test.pdf", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

doc2txt

Description:

Method	Return values	Description
doc2txt(FileStream fs, String fileName)	String	Extracts the text from the uploaded document.
 A Text Extractor must be enabled for the document MIME type on the OpenKM server for it to run.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("C:\\Documents\\test.docx", FileMode.Open);
                System.Console.WriteLine(ws.conversion.doc2txt(fileStream, "test.docx"));
                fileStream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

img2txt

Description:

Method	Return values	Description
img2txt(FileStream fs, String fileName)	String	Extracts the text from the uploaded image.
 The OCR engine must be enabled on the OpenKM server for it to run.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("C:\\Images\\test.png", FileMode.Open);
                System.Console.WriteLine(ws.conversion.img2txt(fileStream, "test.png"));
                fileStream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

barcode2txt

Description:

Method	Return values	Description
barcode2txt(FileStream fs, String fileName)	String	Extracts the barcode text from the uploaded image.
 The Barcode tool must be enabled on the OpenKM server for it to run.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```
        try
        {
            ws.login(user, password);
            FileStream fileStream = new FileStream("C:\\Images\\test.png", FileMode.Open);
            System.Console.WriteLine(ws.conversion.barcode2txt(fileStream, "test.png"));
            fileStream.Close();
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

Dashboard samples

Basic

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web services, the session is kept in the web service object. Then you can use the other API methods.

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

```
ws.dashboard.getUserCheckedOutDocuments();
```

Methods

getUserCheckedOutDocuments

Description:

Method	Return values	Description
getUserCheckedOutDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the documents being edited by the user.
 The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. • The parameter "limit" is used to limit the number of results returned. • The parameter "offset" tells the API to skip that many results before beginning to return results.		
 For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:		

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserCheckedOutDocuments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLastModifiedDocuments

Description:

Method	Return values	Description
getUserLastModifiedDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the most recently modified documents by the user.



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

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastModifiedDocuments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLockedDocuments

Description:

Method	Return values	Description
getUserLockedDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the documents locked by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLockedDocuments(0, 5);
            }
        }
    }
}

```

```

        foreach (DashboardResult dashboardResult in resultSet.results)
        {
            System.Console.WriteLine(dashboardResult.node.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getUserLockedRecords

Description:

Method	Return values	Description
getUserLockedRecords(int offset, int limit)	DashboardResultSet	Returns a list of the records locked by the user.

✓ The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" tells the API to skip that many results before 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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{

```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8180/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedRecords(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getUserLockedFolders

Description:

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



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

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLockedFolders(0,5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLockedMails

Description:

Method	Return values	Description
getUserLockedMails(int offset, int limit)	DashboardResultSet	Returns a list of the mails locked by the user.
 The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. • The parameter "limit" is used to limit the number of results returned. • The parameter "offset" tells the API to skip that many results before beginning to return results.		
 For example, if your query has 1000 results, but you only want to return the first 10, you should use these values: • limit=10		

- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLockedMails(0,5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserSubscribedDocuments

Description:

Method	Return values	Description
getUserSubscribedDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the documents subscribed by the user.



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

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserSubscribedDocuments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserSubscribedFolders

Description:

Method	Return values	Description
getUserSubscribedFolders(int offset, int limit)	DashboardResultSet	Returns a list of the folders subscribed by the user.



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

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserSubscribedFolders(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {

```

```

        System.Console.WriteLine(dashboardResult.node.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

getUserSubscribedRecords

Description:

Method	Return values	Description
getUserSubscribedRecords(int offset, int limit)	DashboardResultSet	Returns a list of the records subscribed by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program

```

```

{
    static void Main(string[] args)
    {
        String host = "http://localhost:8180/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedRecords(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getUserLastCreatedDocuments

Description:

Method	Return values	Description
getUserLastCreatedDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the last documents created by the user.



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

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedDocuments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

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.		
 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastDocumentsNotesCreated(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedFolders(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLastFoldersNotesCreated

Description:

Method	Return values	Description
getUserLastFoldersNotesCreated(int offset, int limit)	DashboardResultSet	Returns a list of the last folders 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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastFoldersNotesCreated(0, 5);
            }
        }
    }
}

```

```

        foreach (DashboardResult dashboardResult in resultSet.results)
        {
            System.Console.WriteLine(dashboardResult.node.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest

```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedRecords(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. • The parameter "limit" is used to limit the number of results returned. • The parameter "offset" tells the API to skip that many results before beginning to return results.		
 For example, if your query has 1000 results, but you only want to return the first 10, you should use these values: • limit=10 • offset=0 Now suppose you want to show the results from 11-20, you should use these values: • limit=10 • offset=10		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastRecordsNotesCreated(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLastDownloadedDocuments

Description:

Method	Return values	Description
getUserLastDownloadedDocuments(int offset, int limit)	DashboardResultSet	Returns a list of the last documents downloaded by the user.
 The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. - The parameter "limit" is used to limit the number of results returned. - The parameter "offset" tells the API to skip that many results before beginning to return results.		
For example if your query has 1000 results, but you only want to return the first 10, you should use these		



values:

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastDownloadedDocuments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getUserLastImportedMails

Description:

Method	Return values	Description
getUserLastImportedMails(int offset, int	DashboardResultSet	Returns a list of the last mails imported 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMails(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

```
}
}
```

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.

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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
try
{
    ws.login(user, password);
    DashboardResultSet resultSet = ws.dashboard.getUserLastMailsNotesCreated(0, 5);
    foreach (DashboardResult dashboardResult in resultSet.results)
    {
        System.Console.WriteLine(dashboardResult.node.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
```

getUserLastImportedMailAttachments

Description:

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



The parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMailAttachments(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getUserSearches

Description:

Method	Return values	Description
getUserSearches()	List<QueryParams>	Returns a list of the searches saved by the user as user news.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try

```

```

    {
        ws.login(user, password);
        List<QueryParams> results = ws.dashboard.getUserSearches();
        foreach (QueryParams userSearch in results)
        {
            System.Console.WriteLine(userSearch.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

findUserSearches

Description:

Method	Return values	Description
findUserSearches(long qpId)	List<DashboardNodeResult>	Returns a list of nodes based on a search saved by the user (search of type user news).

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<DashboardNodeResult> results = ws.dashboard.findUserSearches();
                foreach (DashboardNodeResult nodeResult in results)
                {
                    System.Console.WriteLine(nodeResult.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

Document samples

Basics

On most methods, you'll see the parameter named "**docId**" and "**fldId**". The values of these parameters can be valid document and folder **UUIDs** respectively.



Example of docId:

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

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the "**login**" method. You can access the "**login**" method on the webservice object "**ws**" as shown below:

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

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

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

Methods

create

Description:

Method	Return values	Description
create(String fldId, FileInfo fi)	Document	Creates a new document and returns an object of type Document.
Parameters:		

- **fldId** is the UUID of the folder where the document will be created.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileInfo fileInfo = new FileInfo("E:\\doc.docx");
                ws.document.create(fldId, fileInfo);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

create

Description:

Method	Return values	Description
create(String uuid, String name, FileStream fs, long nodeClass)	Document	Creates a new document with nodeClass. Returns a Document object with the properties of the created document.
The values of the uuid parameter should be a folder or record node UUID. The nodeClass parameter should be a valid business type (series) value. A nodeClass value of 0 means there's no business type (series) selected.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("E:\\logo.png", FileMode.Open);
                long nodeClass = 0;
                ws.document.create("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43", "logo.png", fileStream, nodeClass);
                fileStream.Dispose();
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

create

Description:

Method	Return values	Description
create(String fldId, String name, FileStream fs)	Document	Creates a new document and returns an object of type Document.
Parameters: • fldId is the UUID of the folder where the document will be created. • name is the name of the document including the extension.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;

```

```

using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("E:\\logo.png", FileMode.Open);
                ws.document.create(fldId, "logo.png", fileStream);
                fileStream.Dispose();
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

delete

Description:

Method	Return values	Description
delete(String docId)	void	Deletes a document.



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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    ws.document.delete("f123a950-0329-4d62-8328-0ff500fd42db");
} catch (Exception e) {
    System.Console.WriteLine(e.ToString());
}

System.Console.ReadKey();
}
}
}

```

getProperties

Description:

Method	Return values	Description
getProperties(String docId)	Document	Returns the document properties.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.getProperties("f123a950-0329-4d62-8328-0ff500fd42db").toString());
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getContent

Description:

Method	Return values	Description
getContent(String docId)	Stream	Retrieves a document's content (binary data) of the current document version.



If you send the file across a Servlet response, we suggest setting the content length with:

```
Document doc = ws.getDocumentProperties(docId);
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

 We've found incorrect size problems while using:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream tmpFile = ws.document.getContent("f123a950-0329-4d62-8328-0ff500fd42db");
                FileStream destFile = new FileStream("C:\\test.png", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

}

getContentByVersion

Description:

Method	Return values	Description
getContentByVersion(String docId,String versionId)	Stream	Retrieves a document's content (binary data) of a specific document version.



If you send the file across a Servlet response, we suggest setting the content length with:

```
Document doc = ws.getDocumentProperties(docId);
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

 We've found incorrect size problems while using:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream tmpFile = ws.document.getContentByVersion("f123a950-0329-4d62-8328-0ff500fd42db","1.1");
                FileStream destFile = new FileStream("C:\\test.png", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
            } catch (Exception e) {
```

```
        System.Console.WriteLine(e.ToString());
    }
}
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String fldId)	List<Document>	Returns a list of all documents whose parent is fldId.
The parameter fldId can be a folder or a record node.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Document doc in ws.document.getChildren("f123a950-0329-4d62-8328-0ff500fd42db"))
                {
                    System.Console.WriteLine(doc);
                }
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

rename

Description:

Method	Return values	Description

rename(String docId, String newName)	Document	Changes the name of a document.
---	-----------------	---------------------------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Document doc = ws.document.rename("f123a950-0329-4d62-8328-0ff500fd42db", "new_name.png");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setProperties

Description:

Method	Return values	Description
setProperties(String docId, String title, String description, String lang, List<String> keywords, List<String> categories)	void	Changes some document properties.
Variables allowed to be changed: • Title • Description • Language • Associated categories • Associated keywords		



Only non-null and non-empty variables will be taken into consideration.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<string> keywords = new List<string>();
                keywords.Add("testA");
                keywords.Add("testB");
                List<string> categories = new List<string>();
                categories.Add("/okm:categories");
                ws.document.setProperties("f123a950-0329-4d62-8328-0ff500fd42db", "anyTitle", "anyDescription", "es-E
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setNodeClass

Description:

Method	Return values	Description
setNodeClass(String docId, String ncId)	void	Sets the document node class ID.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setNodeClass("f123a950-0329-4d62-8328-0ff500fd42db",ncID);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

checkout

Description:

Method	Return values	Description
checkout(String docId)	void	Marks the document for editing.
Only one user can modify a document at the same time. Before starting editing, you must perform a check-out action that locks the editing process for other users and allows only the user who executed the action.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        ws.document.checkout("f123a950-0329-4d62-8328-0ff500fd42db");
        // At this point the document is locked for other users except for the user who executed the action
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

cancelCheckout

Description:

Method	Return values	Description
cancelCheckout(String docId)	void	Cancels a document edition.



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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(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("f123a950-0329-4d62-8328-0ff500fd42db");
                // At this point other users are allowed to execute a checkout and modify the document
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

forceCancelCheckout

Description:

Method	Return values	Description
forceCancelCheckout(String docId)	void	Cancels a document edition.

This method allows canceling the edition of any document.

It is not mandatory for the same user who previously executed the checkout action to perform this action.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(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("f123a950-0329-4d62-8328-0ff500fd42db");
                // At this point other users are allowed to execute a checkout and modify the document
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

isCheckedOut

Description:

Method	Return values	Description
isCheckedOut(String docId)	Boolean	Returns a boolean indicating whether the document is being edited.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Is the document checkout:" + ws.document.isCheckedOut("f123a950-0329-4d
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

checkin

Description:

Method	Return values	Description
checkin(String uuid, FileStream fs, String comment)	Version	Updates a document with a new version and returns an object with the new Version values.
 Only the user who started the edition (checkout) is allowed to update the document.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;

namespace OKMRest
{
```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\logo.png", FileMode.Open);
            ws.document.checkin("f123a950-0329-4d62-8328-0ff50fd42db", fs, "optional some comment");
            fs.Dispose();
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

checkin

Description:

Method	Return values	Description
checkin(String uuid, FileStream fs, String comment, int increment)	Version	Updates a document with a new version and returns an object with new Version values.
 The value of the increment variable must be 1 or greater. The valid values of the increment variable depend on the VersionNumberAdapter you have enabled.  Only the user who started the edition - checkout - is allowed to update the document.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    FileStream fs = new FileStream(@"C:\Desktop\logo.png", FileMode.Open);
    ws.document.checkin("f123a950-0329-4d62-8328-0ff500fd42db", fs, "optional some comment", 1);
    fs.Dispose();
} catch (Exception e) {
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

purge

Description:

Method	Return values	Description
purge(String docId)	void	The document is permanently removed from the repository.

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



When a document is purged it can only be restored from a previous repository backup. The purge action removes the document permanently from the repository.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.purge("f123a950-0329-4d62-8328-0ff500fd42db");
            }
        }
    }
}

```

```
    } catch (Exception e) {  
        System.Console.WriteLine(e.ToString());  
    }  
}  
}
```

move

Description:

Method	Return values	Description
move(String docId, String dstId)	void	Moves a document into a folder or record.
The values of the dstId parameter should be a folder or record UUID or path.		

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                ws.document.move("f123a950-0329-4d62-8328-0ff500fd42db", fldId);  
            } catch (Exception e) {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

copy

Description:

Method	Return values	Description
copy(String uuid, String dstId, String newName)	Document	Copies a document into a folder or record.

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

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



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

See the "**extendedCopy**" method for this feature.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.copy("f123a950-0329-4d62-8328-0ff500fd42db", "0ff500fd42db-0329-4d62-8328-f123a950")
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getVersionHistorySize

Description:

Method	Return values	Description
getVersionHistorySize(String docId)	long	Returns the total size in bytes of all documents in the document history.

Example:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long byteCount = ws.document.getVersionHistorySize(docUUId);
                string[] suf = { "B", "KB", "MB", "GB", "TB", "PB", "EB" }; //Longs run out around EB
                long bytes = Math.Abs(byteCount);
                int place = Convert.ToInt32(Math.Floor(Math.Log(bytes, 1024)));
                double num = Math.Round(bytes / Math.Pow(1024, place), 1);
                System.Console.WriteLine((Math.Sign(byteCount) * num).ToString() + suf[place]);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

isValid

Description:

Method	Return values	Description
isValid(String docId)	Boolean	Returns a boolean indicating whether the node is a document.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try

```

```

    {
        ws.login(user, password);
        // Return true
        ws.document.isValid("f123a950-0329-4d62-8328-0ff500fd42db");

        // Return false
        ws.document.isValidDocument("/okm:root");
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getPath

Description:

Method	Return values	Description
getPath(String docId)	String	Converts a document UUID to a document path.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.getPath("e339f14b-4d3a-489c-91d3-05e4575709d2"));
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

extendedCopy

Description:

Return

Method	values	Description
extendedCopy(String nodeId, String dstId, String newName, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)	void	Copies a document with associated data into a 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 newName parameter value is null, the document will preserve the same name.



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

Additional:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
```

```
        ws.login(user, password);
        ws.document.extendedCopy("f123a950-0329-4d62-8328-0ff500fd42db", "d1cad1b0-3c4f-4ad3-8ee1-6ad7");
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

getExtractedText

Description:

Method	Return values	Description
getExtractedText(String docId)	String	Returns a String with the text extracted by the text extraction process.

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



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

Despite this restriction, database queries can be performed via the API by administrator users to retrieve this information. Check [Repository samples](#) for accessing database-level information.



More information at [Statistics](#).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
```

```
        ws.login(user, password);
        System.Console.WriteLine(ws.document.getExtractedText("f123a950-0329-4d62-8328-0ff500fd42db"));
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

setExtractedText

Description:

Method	Return values	Description
setExtractedText(String uuid, FileStream fileStream)	void	Sets the extracted text.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream(@"C:\test.pdf", FileMode.Open, FileAccess.Read);
                ws.document.setExtractedText("f123a950-0329-4d62-8328-0ff500fd42db", fs);
                fs.Close();
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getThumbnail

Description:

Method	Return values	Description
getThumbnail(String docId, ThumbnailType type)	Stream	Returns a 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream st = ws.document.getThumbnail("f123a950-0329-4d62-8328-0ff500fd42db", ThumbnailType.THUMBNAIL_SEARCH);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setLanguage

Description:

Method	Return values	Description
setLanguage(String docId, String lang)	void	Sets a document language.
The parameter lang must be ISO 639-1 compliant.		



More information at https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setLanguage("f123a950-0329-4d62-8328-0ff500fd42db", "es-ES");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(String docId, String title)	void	Sets a document title.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
```

```

    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            ws.document.setTitle("f123a950-0329-4d62-8328-0ff500fd42db", "Some title here");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

createFromTemplate

Description:

Method	Return values	Description
createFromTemplate(String uuid, String dstPath , Boolean categories, Boolean keywords, Boolean notes, Boolean security, Dictionary<String, String> properties)	Document	Creates a new document from a 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, these values are mandatory and must be set in the properties parameter. i For more information about templates and metadata, take a look at Creating templates. i Additional: • When the categories parameter is true, the original values of the categories will be copied. • When the keywords parameter is true, the original values of the keywords will be copied. • When the notes parameter is true, the original values of the notes will be copied.		

Example:



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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("okp:tpl.name", "Some name");
                DateTime date = DateTime.Now;
                // Value must be converted to String ISO 8601 compliant
                properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
                properties.Add("okp:tpl.language", "java");
                ws.document.createFromTemplate("f123a950-0329-4d62-8328-0ff500fd42db", "/okm:root/test.pdf", false,
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

updateFromTemplate

Description:

Method	Return values	Description
updateFromTemplate(String uuid, String dstId, Map<String, String> properties)	Document	Updates a document previously created from a template and returns a Document object.
The uuid value is the UUID value of the template file. The dstId is the UUID of the document to be updated. This method only makes sense when the template uses metadata groups to fill fields.		



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

Example:



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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("okp:tpl.name", "Some name");
                DateTime date = DateTime.Now;
                // Value must be converted to String ISO 8601 compliant
                properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
                properties.Add("okp:tpl.language", "java");

                ws.document.updateFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "058b9379-b441-454d-85
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getDetectedLanguages

Description:

Method	Return values	Description
getDetectedLanguages()	List<String>	Returns a list of available document languages that OpenKM can identify.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (String lang in ws.document.getDetectedLanguages())
                {
                    System.Console.WriteLine(lang);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getAnnotations

Description:

Method	Return values	Description
getAnnotations(String docId, String verName)	String	Returns the document annotations for a document version.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            System.Console.WriteLine(ws.document.getAnnotations("2b9721e7-f186-4eea-ad61-42c97353548f", "1.0");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getDifferences

Description:

Method	Return values	Description
getDifferences(String docId, String v1, String v2)	Stream	Returns a PDF document with the differences between two document versions.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                BeanHelper beanHelper = new BeanHelper();
                Stream stream = ws.document.getDifferences("22799885-973f-46dc-a82c-dd11d2c65c65", "1.0", "1.1");
            }
        }
    }
}

```

```

        Byte[] data = beanHelper.ReadToEnd(stream);
        FileStream fileStream = new FileStream(@"C:\Desktop\out.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
        foreach (byte b in data)
        {
            fileStream.WriteByte(b);
        }
        fileStream.Close();
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getCheckedOut

Description:

Method	Return values	Description
getCheckedOut()	List<Document>	Returns a list of documents checked out by the user.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Document> docs = ws.document.getCheckedOut();
                foreach (Document doc in docs)
                {
                    System.Console.WriteLine(doc.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setDescription

Description:

Method	Return values	Description
setDescription(String docId, String description)	void	Sets a document description.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setDescription("f123a950-0329-4d62-8328-0ff500fd42db", "Some description here");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(String docId, long stage)	void	Sets a document disposition stage.

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long stage = 1;
                ws.document.setDispositionStage("f123a950-0329-4d62-8328-0ff500fd42db", stage);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

createWizard

Description:

Method	Return values	Description
createWizard(String docId, long stage)	WizardNode	Creates a document with a wizard.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        FileStream fs = new FileStream(LOCAL_DOC, FileMode.Open, FileAccess.Read);
        long nodeClass= 1;
        WizardNode wizardNode = ws.document.createWizard("f123a950-0329-4d62-8328-0ff500fd42db","test.doc");
        fs.Close();
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

isOCRDataCaptureSupported

Description:

Method	Return values	Description
isOCRDataCaptureSupported(String docId)	boolean	Checks if the node supports OCR data capture.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                if(ws.document.isOCRDataCaptureSupported("f123a950-0329-4d62-8328-0ff500fd42db"))
                {
                    System.Console.WriteLine("Yes supported OCR Data capture");
                }
                else
                {
                    System.Console.WriteLine("No supported OCR Data capture");
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
```

recognize

Description:

Method	Return values	Description
recognize(String docId)	OCRRecognise	Returns an OCRRecognise object.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                OCRRecognise ocr = ws.document.recognize("f123a950-0329-4d62-8328-0ff500fd42db");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

captureData

Description:

Method	Return values	Description
captureData(String docId, long templateId)	void	Proceeds with the OCR data capture using the OCR template identified by templateId.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long templateId = 1;
                ws.document.captureData("f123a950-0329-4d62-8328-0ff500fd42db", templateId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getNumberOfPages

Description:

Method	Return values	Description
getNumberOfPages(String docId)	int	Gets the number of pages of a document.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    System.Console.WriteLine("The number of pages is : " + ws.document.getNumberOfPages("f123a950-0329-4d62-8328-0ff500fd42db", 1));
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

getPageAsImage

Description:

Method	Return values	Description
getPageAsImage(String uuid, int pageNumber, int maxWidth, int maxHeight)	String	Returns the specified page as an image encoded as a Base64 string.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.getPageAsImage("f123a950-0329-4d62-8328-0ff500fd42db", 1));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

mergePdf

Description:

Method	Return values	Description
mergePdf(String destinationUuid, String docName, List<String> uuids)	String	Merges two or more PDF files.
The parameter destinationUuid should be a valid folder or record UUID.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<string> uuids = new List<string>();
                uuids.Add("a978f8e6-aa87-4f0f-b7bc-6f44cb090fdf");
                uuids.Add("56a6d75f-563e-4ccd-ace6-d07d7860bae5");
                string folderUuid = "15e35aef-f1bb-451e-ac86-3b0f7c88ca9f";
                string uuid = ws.document.mergePdf(folderUuid, "MergePdf.pdf", uuids);
                Document mergePdf = ws.document.getDocumentProperties(uuid);
                Console.WriteLine(mergePdf.path);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getLiveEditRestrictedMimeTypes

Description:

Method	Return values	Description
getLiveEditRestrictedMimeTypes()	List<String>	Returns a list of MIME types that cannot be used with the live edit feature.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<String> mymTypes = ws.document.getLiveEditRestrictedMimeTypes();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

liveEditCheckin

Description:

Method	Return values	Description
liveEditCheckin(String uuid, String comment, int increment)	void	Performs a checkin.



This method should be executed only if the document has been edited with the live edit feature.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.liveEditCheckin("8b9a32c8-db65-41c8-a2a0-af03761f60b6", "Any comment", 1);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

liveEditCancelCheckout

Description:

Method	Return values	Description
liveEditCancelCheckout(String uuid)	void	Cancels a document edit.
 This method should be executed only if the document has been edited with the live edit feature.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    // At this point the document is locked for other users except for the user who executed the action
    ws.document.liveEditCancelCheckout("f123a950-0329-4d62-8328-0ff500fd42db");
    // At this point other users are allowed to execute a checkout and modify the document
} catch (Exception e) {
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

liveEditForceCancelCheckout

Description:

Method	Return values	Description
liveEditForceCancelCheckout(String uuid)	void	Cancel a document edition.



This method should be executed only if the document has been edited with the live edit feature.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // At this point the document is locked for other users except for the user who executed the action
                ws.document.liveEditForceCancelCheckout("f123a950-0329-4d62-8328-0ff500fd42db");
                // At this point other users are allowed to execute a checkout and modify the document
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

    }
  }
}

```

liveEditSetContent

Description:

Method	Return values	Description
liveEditSetContent(String uuid, FileStream fs	void	Updates the content of the document edited with the live edit feature.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream(@"D:\Test.pdf", FileMode.Open, FileAccess.Read);
                ws.document.liveEditSetContent("1ec49da9-1746-4875-ae32-9281d7303a62", fs);
                fs.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

IsAttachment

Description:

Method	Return values	Description

isAttachment(String docId)	Boolean	Returns a boolean indicating whether the node is a mail attachment.
-----------------------------------	----------------	---

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.isAttachment("1ec49da9-1746-4875-ae32-9281d7303a62").ToS
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

isConvertibleToPDF

Description:

Method	Return values	Description
isConvertibleToPDF(String docId)	Boolean	Returns a boolean indicating whether the node is convertible to PDF.



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

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.isConvertibleToPDF("1ec49da9-1746-4875-ae32-9281d7303a62"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getPdf

Description:

Method	Return values	Description
getPdf(String uuid)	Stream	Returns a PDF of the document.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        Document doc = ws.document.getDocumentProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
        Stream stream = ws.document.getPdf(doc.uuid);
        FileStream docFile = new FileStream("D:\\test.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
        stream.CopyTo(docFile);
        stream.Close();
        docFile.Close();
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

saveAsPdf

Description:

Method	Return values	Description
saveAsPdf(String uuid, String newName, bool propertyGroups, bool security)	Document	Saves the document as a PDF in the OpenKM repository.
The value of the uuid parameter should be a Document UUID. When the value of the newName parameter is null, the document will preserve its name. i Additional: • When the propertyGroups parameter is true, the metadata groups of the source are copied to the target. • When the security parameter is true, the security of the source is copied to the target.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    Document doc = ws.document.getDocumentProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
    Document docPdf = ws.document.saveAsPdf(doc.uuid, "test.pdf", true, true);
    System.Console.WriteLine("Document pdf: " + docPdf.path);
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

webPageImport

Description:

Method	Return values	Description
webPageImport(String uuid, String url)	void	Saves the content of a web page as a PDF document in the OpenKM repository.
The value of the uuid parameter should be a folder or record node UUID.		
The value of the url parameter should be a valid web page URL.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try

```

```

        {
            ws.login(user, password);
            ws.document.webPageImport("271cc3aa-218e-4574-8065-c34c704f4a20", "https://www.google.com/");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

stamp

Description:

Method	Return values	Description
stamp(String uuid, int type, long stampId)	void	Stamps the document.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int type=1;
                stampId=1;
                ws.document.stamp("271cc3aa-218e-4574-8065-c34c704f4a20", type, stampId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getXrefs

Description:

Method	Return values	Description
getXrefs(String uuid)	List<XRef>	Returns a list of document references for a document.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (XRef xRef in ws.document.getXrefs("5ce6dd37-7472-404a-878e-274005a2d6db"))
                {
                    Console.WriteLine(xRef.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

refresh

Description:

Method	Return values	Description
refresh(String uuid)	void	Refreshes the references of a document.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.refresh("5ce6dd37-7472-404a-878e-274005a2d6db");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

FilePlan samples

Basic

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in to the web services, the session is kept in the web service object. Then you can use the other API methods.

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

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

Methods

getRootSections

Description:

Method	Return values	Description
getRootSections()	List<NodeClassSectionDefinition>	Returns a list of the NodeClassSectionDefinition.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
```

```

    {
        String host = "http://localhost:8180/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            List<NodeClassSectionDefinition> results = ws.filePlan.getRootSections();
            foreach (NodeClassSectionDefinition ncsd in results)
            {
                System.Console.WriteLine(ncsd.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getRootSectionsFilteredBySecurity

Description:

Method	Return values	Description
getRootSectionsFilteredBySecurity(int permissions)	List<NodeClassSectionDefinition>	Returns a list of the NodeClassSectionDefinition by permissions.



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

    try
    {
        ws.login(user, password);
        List<NodeClassSectionDefinition> results = ws.filePlan.getRootSectionsFilteredBySecurity(permissions);
        foreach (NodeClassSectionDefinition ncsd in results)
        {
            System.Console.WriteLine(ncsd.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getChildrenSections

Description:

Method	Return values	Description
getChildrenSections(long parentId)	List<NodeClassSectionDefinition>	Returns a list of the NodeClassSectionDefinition by parent section.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long parentId = 1;
                List<NodeClassSectionDefinition> results = ws.filePlan.getChildrenSections(parentId);
                foreach (NodeClassSectionDefinition ncsd in results)
                {
                    System.Console.WriteLine(ncsd.toString());
                }
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```

        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getChildrenSectionsFilteredBySecurity

Description:

Method	Return values	Description
getChildrenSectionsFilteredBySecurity(long parentId, int permissions)	List<NodeClassSectionDefinition>	Returns a list of the NodeClassSectionDefinition by parent section and permissions.



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long parentId = 1;
                int permissions = Permission.ALL_GRANTS;
                List<NodeClassSectionDefinition> results = ws.filePlan.getChildrenSectionsFilteredBySecurity(parentId, permissions);
                foreach (NodeClassSectionDefinition ncscd in results)
                {
                    System.Console.WriteLine(ncscd.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}  
}
```

getChildrenClasses

Description:

Method	Return values	Description
getChildrenClasses(long sectionId)	List<NodeClass>	Returns a list of the NodeClass by section.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8180/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long sectionId = 1;  
                List<NodeClass> results = ws.filePlan.getChildrenClasses(sectionId);  
                foreach (NodeClass nc in results)  
                {  
                    System.Console.WriteLine(nc.toString());  
                }  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getChildrenClassesFilteredBySecurity

Description:

Method	Return values	Description

getChildrenClassesFilteredBySecurity(long sectionId, int permissions)	List<NodeClass>	Returns a list of the NodeClass by section and permissions.
--	------------------------------	---



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long sectionId = 1;
                int permissions = Permission.ALL_GRANTS;
                List<NodeClass> results = ws.filePlan.getChildrenClassesFilteredBySecurity(sectionId, permissions);
                foreach (NodeClass nc in results)
                {
                    System.Console.WriteLine(nc.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findNodeClassByPk

Description:

Method	Return values	Description
findNodeClassByPk(long id)	NodeClass	Returns a NodeClass.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = 0;
                NodeClass nodeClass= ws.filePlan.findNodeClassByPk(id);
                System.Console.WriteLine(nodeClass.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findElectronicRecordClasses

Description:

Method	Return values	Description
findElectronicRecordClasses()	List<NodeClass>	Returns a list of the NodeClass that are Electronic Records.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
```

```

String host = "http://localhost:8180/openkm";
String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    List<NodeClass> results = ws.filePlan.findElectronicRecordClasses();
    foreach (NodeClass nc in results)
    {
        System.Console.WriteLine(nc.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

findElectronicRecordClassesFilteredBySecurity

Description:

Method	Return values	Description
findElectronicRecordClassesFilteredBySecurity(int permissions)	List<NodeClass>	Returns a list of the NodeClass by section and permissions that are Electronic Records.



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

    try
    {
        ws.login(user, password);
        int permissions = Permission.ALL_GRANTS;
        List<NodeClass> results = ws.filePlan.findElectronicRecordClassesFilteredBySecurity(permissions);
        foreach (NodeClass nc in results)
        {
            System.Console.WriteLine(nc.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

findFilteredByCodeOrNameFilteredBySecurity

Description:

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



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String code = string.Empty;
                String name = "test";
            }
        }
    }
}

```

```

        int permissions = Permission.ALL_GRANTS;
        List<NodeClass> results = ws.filePlan.findFilteredByCodeOrNameFilteredBySecurity(code, name, permissions);
        foreach (NodeClass nc in results)
        {
            System.Console.WriteLine(nc.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

findSectionFiltered

Description:

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



The class **com.openkm.sdk4csharp.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get security grants.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String code = string.Empty;
                String name = "test";
                int permissions = Permission.ALL_GRANTS;
                List<NodeClassSectionDefinition> results = ws.filePlan.findSectionFiltered(code, name, permissions);
            }
        }
    }
}

```

```

        foreach (NodeClassSectionDefinition ncsd in results)
        {
            System.Console.WriteLine(ncsd.ToString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getNodeClassBreadcrumb

Description:

Method	Return values	Description
getNodeClassBreadcrumb(long sectionId)	List<NodeClassSectionDefinition>	Returns a list of the NodeClassSectionDefinition that are Electronic Records.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long sectionId = 1;
                List<NodeClassSectionDefinition> results = ws.filePlan.getNodeClassBreadcrumb(sectionId);
                foreach (NodeClassSectionDefinition ncsd in results)
                {
                    System.Console.WriteLine(ncsd.ToString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

findAllNodeClasses

Description:

Method	Return values	Description
findAllNodeClasses()	List<NodeClass>	Returns a list of the NodeClass.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8180/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                List<NodeClass> results = ws.filePlan.findAllNodeClasses();  
                foreach (NodeClass nc in results)  
                {  
                    System.Console.WriteLine(nc.toString());  
                }  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getRootSectionsIdWithChildren

Description:

Method	Return values	Description
getRootSectionsIdWithChildren()	List<long>	Returns a list of the section ids that have children and whose

	parent is root.
--	-----------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<long> results = ws.filePlan.getRootSectionsIdWithChildren();
                foreach (var id in results)
                {
                    System.Console.WriteLine(id.ToString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getChildrenSectionsIdByTenantWithChildren

Description:

Method	Return values	Description
getChildrenSectionsIdByTenantWithChildren(long parentId)	List<long>	Returns a list of the section ids that have children with parent equal to parentId and are filtered by tenant.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long parentId = 1;
                List<long> result = ws.filePlan.getChildrenSectionsIdByTenantWithChildren(parentId);
                foreach (var id in result)
                {
                    System.Console.WriteLine(id.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Folder samples

Basics

On most methods, you'll see the parameter named "**fldId**". The value of this parameter can be a valid folder **UUID**.



Example of fldId:

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

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web service, the session is kept in the web service object. Then you can use the other API methods.

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

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

Methods

create

Description:

Method	Return values	Description
create(String fldId, String name)	Folder	Creates a new folder and returns a Folder object.

Example:

```
using System;  
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.create("f123a950-0329-4d62-8328-0ff500fd42db", "test");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

create

Description:

Method	Return values	Description
create(String uuid, String name, long nodeClass)	Folder	Creates a new folder and returns a Folder object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        long nodeClass = 0;
        ws.folder.create("f123a950-0329-4d62-8328-0ff500fd42db", "test", nodeClass);
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getProperties

Description:

Method	Return values	Description
getProperties(String fldId)	Folder	Returns the folder properties.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.folder.getProperties("f123a950-0329-4d62-8328-0ff500fd42db").toString());
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

delete

Description:

Method	Return values	Description
delete(String fldId)	void	Deletes a folder.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.delete("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(String fldId, String newName)	Folder	Renames a folder.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```

        try
        {
            ws.login(user, password);
            // Exists folder /okm:root/test
            ws.folder.rename("f123a950-0329-4d62-8328-0ff500fd42db", "renamedFolder");
            // Folder has renamed to /okm:root/renamedFolder
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

move

Description:

Method	Return values	Description
move(String fldId, String dstId)	void	Moves a folder into another folder or record.
The values of the dstId parameter should be a folder or record UUID or a path.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Exists folder /okm:root/test
                ws.folder.move("f123a950-0329-4d62-8328-0ff500fd42db", "ac165cab-a62a-4b17-89fc-2480a1d784db");
                // Folder has moved to /okm:root/tmp/test
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getChildren

Description:

Method	Return values	Description
getChildren(String fldId)	List<Folder>	Returns a list of all folders whose parent is fldId.
The parameter fldId can be a folder or a record node.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Folder fld in ws.folder.getChildren("f123a950-0329-4d62-8328-0ff500fd42db"))
                {
                    System.Console.WriteLine(fld.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

isValid

Description:

Method	Return values	Description
isValid(String fldId)	Boolean	Returns a boolean that indicates whether the node is a folder.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Return true
                ws.folder.isValid("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getPath

Description:

Method	Return values	Description
getPath(String uuid)	String	Converts a folder UUID to a folder path.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);

```

```

        System.Console.WriteLine(ws.folder.getPath("f123a950-0329-4d62-8328-0ff500fd42db"));
    } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

copy

Description:

Method	Return values	Description
copy(String fldId, String dstId, String newName)	Folder	Copies a folder into a folder or record.
The values of the dstId parameter should be a folder or record UUID or a path. When the newName parameter value is null, the folder will preserve the same name.  Only the security grants are copied to the destination. The folder's metadata, keywords, etc. are not copied. See the "extendedFolderCopy" method for this feature.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.copy("f123a950-0329-4d62-8328-0ff500fd42db", "a321a950-0329-4d62-8328-0ff500fd42db", "newFolder");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(String fldId, String dstId, String newName, bool categories, bool keywords, bool propertyGroups, bool notes, bool wiki, bool security)	Folder	Copies a folder with associated data into another folder or record.

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

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



By default, only the binary data and the security grants are copied; the folder's metadata, keywords, etc. 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 property groups parameter is true, the original values of the metadata groups will be copied.
- When the notes parameter is true, the original values of the notes will be copied.
- When the wiki parameter is true, the original values of the wiki will be copied.
- When the security parameter is true, the original security settings will be copied.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
        }
    }
}
```

```

        try
        {
            ws.login(user, password);
            ws.folder.extendedCopy("f123a950-0329-4d62-8328-0ff500fd42db", "055e4384-7c70-4456-b32b-f5a55a7
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getContentInfo

Description:

Method	Return values	Description
getContentInfo(String fldId)	ContentInfo	Returns a ContentInfo object with information about the folder.
The ContentInfo object retrieves information about: • The number of folders. • The number of documents. • The number of records. • The number of emails. • The size in bytes of all objects in the folder.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.folder.getContentInfo(fldUUid).toString());
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

    }
  }
}

```

purge

Description:

Method	Return values	Description
purge(String fldId)	void	The folder is permanently removed from the repository.

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



When a folder is purged, it can only be restored from a previous repository backup. The purge action removes the folder permanently from the repository.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.purge("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setStyle

Description:

Method	Return values	Description
setStyle(String fldId, long styleId)	void	Sets the folder style.


[More information at Folder style.](#)

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.setStyle("f123a950-0329-4d62-8328-0ff500fd42db", 1);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createMissingFolders

Description:

Method	Return values	Description
createMissingFolders(String fldPath)	void	Creates missing folders.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.createMissingFolders("/okm:root/missingfld1/missingfld2/missingfld3");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setDescription

Description:

Method	Return values	Description
setDescription(string fldId, String description)	void	Sets the folder description.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.setDescription("f123a950-0329-4d62-8328-0ff500fd42db","Any description");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
```

createFromTemplate

Description:

Method	Return values	Description
createFromTemplate(String uuid, String dstPath, bool categories, bool keywords, bool notes, bool propertyGroups, bool security Dictionary<String, String> properties)	Folder	Creates a new folder from the template and returns a Folder object.
The uuid parameter is the template file. The dstPath can be a folder or a record path. When the template uses metadata groups to fill in fields, these values are mandatory and must be set in the properties parameter. i For more information about Templates and metadata check: Creating templates. i 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 notes parameter is true, the original values of the notes will be copied.		

Example:



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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
```

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

    try
    {
        ws.login(user, password);
        Dictionary<String, String> properties = new Dictionary<String, String>();
        properties.Add("okp:tpl.name", "Some name");
        DateTime date = DateTime.Now;
        // Value must be converted to String ISO 8601 compliant
        properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
        properties.Add("okp:tpl.language", "[\"java\"]");
        Folder foder = ws.folder.createFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "/okm:root/tes
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

Import samples

Basic



We suggest executing the methods below for large imports or for simple creation cases.

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



Example of UUID:

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

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservice the session is kept in the webservice object. Then you can use the other API methods.

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

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

Methods

importDocument

Description:

Method	Return values	Description
importDocument (String uuid, FileInfo fi)	String	Creates a new document and returns the UUID of the document.

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

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileInfo fileInfo = new FileInfo("E:\\doc.docx");
                String uuid = ws.quickImport.importDocument("22c1d190-f798-489d-b420-2008cb38705b", fileInfo);
                System.Console.WriteLine("UUID =" + uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

importDocument

Description:

Method	Return values	Description
importDocument(String uuid, DateTime? date, String author, FileInfo file)	String	Creates a new document and returns the UUID of the document.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```
using System; using System.Collections.Generic;
using System.Text;
```

```

using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileInfo fileInfo = new FileInfo("E:\\doc.docx");
                String uuid = ws.quickImport.importDocument("22c1d190-f798-489d-b420-2008cb38705b", new DateTime(
                System.Console.WriteLine("UUID =" + uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

importFolder

Description:

Method	Return values	Description
importFolder(String uuid, String name)	String	Creates a new folder and returns the UUID of the folder.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            String uuid = ws.quickImport.importFolder("22c1d190-f798-489d-b420-2008cb38705b", "Folder-Name");
            System.Console.WriteLine("UUID =" + uuid);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
}

```

importFolder

Description:

Method	Return values	Description
importFolder(String uuid, String name, DateTime? date, String author)	String	Creates a new folder and returns the UUID of the folder.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String uuid = ws.quickImport.importFolder("22c1d190-f798-489d-b420-2008cb38705b", "Folder-Name", ne
                System.Console.WriteLine("UUID =" + uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

importMail

Description:

Method	Return values	Description
importMail(String uuid, FileInfo fi)	String	Creates a new mail and returns the UUID of the mail.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```
using System; using System.Collections.Generic;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                FileInfo fileInfo = new FileInfo("E:\\test.eml");  
                String uuid = ws.quickImport.importMail("22c1d190-f798-489d-b420-2008cb38705b", fileInfo);  
                System.Console.WriteLine("UUID =" + uuid);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

importRecord

Description:

Method	Return values	Description

importRecord(String uuid, String name)	String	Creates a new record and returns the UUID of the record.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String uuid = ws.quickImport.importRecord("22c1d190-f798-489d-b420-2008cb38705b", "Record-Name");
                System.Console.WriteLine("UUID =" + uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

importRecord

Description:

Method	Return values	Description
importRecord(String uuid, String name, DateTime? date, String author)	String	Creates a new record and returns the UUID of the record.
The value of the uuid parameter should be a folder or record node UUID.		

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String uuid = ws.quickImport.importRecord("22c1d190-f798-489d-b420-2008cb38705b", "Record-Name",
                System.Console.WriteLine("UUID =" + uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Mail samples

Basics

On most methods, you'll see the parameter named "**mailId**". The value of this parameter can be a valid mail **UUID**.



Example of fldId:

- Using UUID -> "50b7a5b9-89d2-430e-bbc9-6a6e01662a71"

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

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

```
ws.mail.getProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbced")
```

Methods

getProperties

Description:

Method	Return values	Description
getProperties(String uuid)	Mail	Returns the mail properties.

Example:

```
using System;  
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.mail.getProperties("f0064cf3-4776-44c2-868a-f08697a6957e").toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

delete

Description:

Method	Return values	Description
delete(String uuid)	void	Deletes a mail.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.delete("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
        }
    }
}

```

```
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

purge

Description:

Method	Return values	Description
purge(String uuid)	void	Folder is permanently removed from the repository.

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

 When a mail is purged, it can only be restored from a previous repository backup. The purge action removes the mail permanently from the repository.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.purge("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(String uuid, String newName)	void	Renames a mail.

 From the OpenKM frontend UI, the subject is used to show the mail name in the file browser table. That means the change will take effect internally on the mail path, but will not be visible in the default UI.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.rename("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "new name");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

move

Description:

Method	Return values	Description
move(String uuid, String dstId)	void	Moves a mail into a folder or record.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.move("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79861f");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

copy

Description:

Method	Return values	Description
public void copy(String mailId, String dstId, String newName)	Mail	Copies mail into a folder or record.
The values of the dstId parameter should be a folder or record UUID. When the newName parameter value is null, the mail will preserve the same name.  Only the security grants are copied to the destination, the metadata, keywords, etc. of the folder are not copied. See "extendedCopy" method for this feature.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.copy("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79861f", "i
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(String mailId, String dstId, bool categories, bool keywords, bool propertyGroups, bool notes, bool security, String newName)	Mail	Copies mail 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 are copied; 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 property groups parameter is true the original values of the metadata groups will be copied.

- When the notes parameter is true the original values of the notes will be copied.
- When the wiki parameter is true the original values of the wiki will be copied.
- When newName is set the mail name is renamed.
- When the security parameter is true the original value of the security will be copied.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.extendedCopy("fe51797c-be0b-4076-8f4b-b4ffe8fd2bc", "055e4384-7c70-4456-b32b-f5a55a7986");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String fldId)	List<Mail>	Returns a list of all mails whose parent is fldId.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Mail mail in ws.mail.getChildren("055e4384-7c70-4456-b32b-f5a55a79861f"))
                {
                    System.Console.WriteLine(mail);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

isValid

Description:

Method	Return values	Description
isValid(String mailId)	Boolean	Returns a boolean that indicates whether the node is a mail.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
            }
        }
    }
}

```

```
        // Return true
        ws.mail.isValid("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

getPath

Description:

Method	Return values	Description
getPath(String mailId)	String	Converts a mail UUID to a folder path.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.mail.getPath("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createAttachment

Description:

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

createAttachment(String mailId, String docName, InputStream is)	Document	Stores an attachment in the mail.
--	-----------------	-----------------------------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream("E:\\logo.png", FileMode.Open);
                ws.mail.createAttachment("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "log.png", fs);
                fs.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

deleteAttachment

Description:

Method	Return values	Description
deleteAttachment(String mailId, String docId)	void	Deletes a mail attachment.
The docId parameter value can be a valid document UUID or path .		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.deleteAttachment("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "e339f14b-4d3a-489c-91d3-05e4
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getAttachments

Description:

Method	Return values	Description
getAttachments(String mailId)	List<Document>	Retrieves a list of all document attachments of a mail.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Document doc in ws.mail.getAttachments("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"))
                {

```

```

        System.Console.WriteLine(doc);
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

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)	Mail	Sends a mail message with an attachment.
The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored in the repository after being sent. Other parameters: • to is a list of destination email addresses. • subject is the mail subject. • cc and bcc are lists of destination email addresses (optional) • docsId is a list of valid document UUIDs already in OpenKM that will be sent as attachments in the mail (optional).		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";

```

```

String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    string from = "test@none.com";
    List<String> to = new List<String>();
    to.Add("destination@mail.com");
    List<String> cc = new List<String>();
    List<String> bcc = new List<String>();
    List<String> docslId = new List<String>();
    List<String> replyTo= new List<String>();
    docslId.Add("7aa523e0-06cf-4733-b8c8-cc74d8b2716d");
    docslId.Add("f123a950-0329-4d62-8328-0ff500fd42db");
    ws.mail.sendMailWithAttachments(from, to, cc, bcc, replyTo, "some subject", "This mail its a test.", docslId);
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

importEml

Description:

Method	Return values	Description
importEml(String dstId, String title, FileStream fs)	Mail	Imports a mail in EML format.
The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored in the repository after being sent.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            FileStream filestream = new FileStream("C:\\Documents\\test.eml", FileMode.Open);
            Mail mail = ws.mail.importEml("f123a950-0329-4d62-8328-0ff500fd42db", "Test", filestream);
            System.Console.WriteLine(mail);
            filestream.Close();
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

importMsg

Description:

Method	Return values	Description
importMsg(String dstId, String title, FileStream fs)	Mail	Imports a mail in MSG format.
The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored in the repository after being sent.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream filestream = new FileStream("C:\\Documents\\test.msg", FileMode.Open);
                Mail mail = ws.mail.importMsg("f123a950-0329-4d62-8328-0ff500fd42db", "Test", filestream);
                System.Console.WriteLine(mail);
                filestream.Close();
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```
        {  
            System.Console.WriteLine(e.ToString());  
        }  
    }  
}
```

setTitle

Description:

Method	Return values	Description
setTitle(String mailId, String title)	Mail	Sets the mail title.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                ws.mail.setTitle("68ed7a93-005c-4ec6-ac01-bb151573c775", "new title");  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

sendMail

Description:

Method	Return values	Description
sendMail(String to, String subject, String body)	Void	Sends a mail.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.sendMail("some@none.com", "Testing sending mail", "Test message body.");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

sendMail

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```

        try
        {
            ws.login(user, password);
            List<String> to = new List<string>();
            to.Add("some@mail.com");
            ws.mail.sendMail(to, "Testing sending mail from OpenKM", "Test message body.");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

sendMail

Description:

Method	Return values	Description
sendMail(String from, List<String> to, String subject, String body)	Void	Sends a mail.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                string from = "test@mome.com"
                List<String> to = new List<string>();
                to.Add("some@mail.com");
                ws.mail.sendMail(from, to, "Testing sending mail from OpenKM", "Test message body.");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setNodeClass

Description:

Method	Return values	Description
setNodeClass(String mailId, long ncId)	void	Sets the mail node class.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.setNodeClass("68ed7a93-005c-4ec6-ac01-bb151573c775", ncId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(String mailId, String description)	void	Sets the mail title.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
```

```
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.setDescription("68ed7a93-005c-4ec6-ac01-bb151573c775", "Any description");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getContent

Description:

Method	Return values	Description
getContent(String mailId)	Stream	Sets the mail title.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream s = ws.mail.getContent("68ed7a93-005c-4ec6-ac01-bb151573c775");
                FileStream mailFile = new FileStream("C:\\mailtest.msg", FileMode.OpenOrCreate);
                s.CopyTo(mailFile);
                s.Close();
                mailFile.Close();
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```

    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

getThumbnail

Description:

Method	Return values	Description
getThumbnail(String mailId, ThumbnailType type)	Stream	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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream s = ws.mail.getThumbnail("68ed7a93-005c-4ec6-ac01-bb151573c775", ThumbnailType.THUMBNAIL_PROPERTIES);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(String uuid, long stage)	void	Sets the mail disposition stage.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long stage = 1;  
                ws.mail.setDispositionStage("f0064cf3-4776-44c2-868a-f08697a6957e", stage);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

createWizard

Description:

Method	Return values	Description
createWizard(String uuid, String title, FileStream fileStream, String type)	WizardNode	Creates a mail with a wizard.



The Wizard node contains the steps that might be followed by the wizard:

- Metadata groups
- Keywords
- Categories

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Folder fld = ws.createFolder("f123a950-0329-4d62-8328-0ff500fd42db", "Test");
                FileStream filestream = new FileStream("C:\\testing.msg", FileMode.Open, FileAccess.Read);
                WizardNode wn = ws.mail.createWizard(fld.uuid, "Any title", filestream, Mail.ORIGIN_MSG);
                filestream.Close();
                System.Console.WriteLine(wn.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getMailsPaginated

Description:

Method	Return values	Description
getMailsPaginated(String context, int offset, int limit, MailFilterQuery filter, String orderColumn, bool orderAsc)	SimpleNodeBaseResultSet	Returns a paginated list of email results filtered by the values of the

MailFilterQuery parameter.



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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning results.
- The parameter "filter" must be the canonical class name of the class which implements the NodeProperties interface.
- The parameter "orderColumn" can have the following values: uuid, subject, from, sentDate and hasAttachments.
- The parameter "orderAsc" can be "true" in case of ascending order and "false" otherwise.

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailFilterQuery filter = new MailFilterQuery();
                filter.subject = "";
            }
        }
    }
}
```

```

        filter.attachments = AttachmentEnum.ALL;
        String orderColumn = "subject";
        SimpleNodeBaseResultSet simpleNodeBaseResultSet = ws.mail.getMailsPaginated(0, 10, filter, orderColumn);

        foreach (SimpleNodeBase simpleNodeBase in simpleNodeBaseResultSet)
        {
            System.Console.WriteLine(simpleNodeBase.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getAccounts

Description:

Method	Return values	Description
getAccounts()	List<MailAccount>	Retrieves a list of user email accounts.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (MailAccount mail in ws.mail.getAccounts())
                {
                    System.Console.WriteLine(mail.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long accountId = 1;
                long start = 1;
                MailServerMessages msm = ws.mail.getMailMessages(accountId, start);
                System.Console.WriteLine(msm.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

addAccount

Description:

Method	Return values	Description
addAccount(MailAccount mailAccount)	void	Add an email account.



It is good practice to create an account and verify the mail configuration with `testMailAccount(MailAccount mail)`.

When you do **not set the user**, the account will be **added by default to the currently logged-in user**.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailAccount mailAccount = new MailAccount();
                mailAccount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
                mailAccount.mailUser = "test@none.com";
                mailAccount.mailPassword = "123456";
                mailAccount.mailHost = "imap.gmail.com";
                mailAccount.mailFolder = "OpenKM";
                mailAccount.active = true;
                mailAccount.recursive = true;
                ws.mail.addAccount(mailAccount);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

updateAccount

Description:

Method	Return values	Description
updateAccount(MailAccount mailAccount)	void	Updates the main configuration data of an email account.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailAccount mailAccount = new MailAccount();
                mailAccount.id = 1; // Valid mail account id;
                mailAccount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
                mailAccount.mailUser = "test@none.com";
                mailAccount.mailPassword = "123456";
                mailAccount.mailHost = "imap.gmail.com";
                mailAccount.mailFolder = "OpenKM";
                mailAccount.active = false;
                mailAccount.mailMarkDeleted = false;
                mailAccount.recursive = false;
                ws.mail.updateAccount(mailAccount);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

testAccount

Description:

Method	Return values	Description
testAccount(MailAccount mailAccount)	void	Verify the connectivity of an email account.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailAccount mailAccount = new MailAccount();
                mailAccount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
                mailAccount.mailUser = "test@none.com";
                mailAccount.mailPassword = "123456";
                mailAccount.mailHost = "imap.gmail.com";
                mailAccount.mailFolder = "OpenKM";
                mailAccount.active = true;
                mailAccount.recursive = true;

                // Test mail account
                ws.mail.testAccount(mailAccount);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

deleteAccount

Description:

Method	Return values	Description
deleteAccount(long mailAccountId)	void	Delete an email account.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    long mailId = 1; //Valid mail id
    ws.mail.deleteAccount(mailId);
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

importMessages

Description:

Method	Return values	Description
importMessages(long mailAccountId, List<long> messageIds)	void	Import messages from a specific email account.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<long> messageIds = new List<long>();
                messageIds.Add(1);
                long mailAccountId = 1; // Valid mail id
                ws.mail.importMessages(mailAccountId, messageIds);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}  
}
```

createFilter

Description:

Method	Return values	Description
createFilter(long mailAccountId, MailFilter mailFilter)	void	Add a filter to an email account.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                Folder fld = ws.createFolder("f123a950-0329-4d62-8328-0ff500fd42db", "FilterFolder");  
                // Valid mail account id  
                long mailAccountId = 1;  
                MailFilter filter = new MailFilter();  
                filter.order = 0;  
                filter.grouping = false;  
                filter.exclusive = true;  
                filter.active = false;  
                filter.node = fld.uuid;  
                ws.mail.createFilter(mailAccountId, filter);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

updateFilter

Description:

Method	Return values	Description
updateFilter(MailFilter mailFilter)	void	Update the filter of an email account.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailFilter filter = new MailFilter();
                filter.id = 1; // Valid filter id
                filter.grouping = true;
                filter.exclusive = false;
                filter.active = true;
                ws.mail.updateFilter(filter);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

deleteFilter

Description:

Method	Return values	Description
deleteFilter(long mailFilterId)	void	Delete a filter from an email account.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long filterId = 1; // Valid filter id
                ws.mail.deleteFilter(filterId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

createRule

Description:

Method	Return values	Description
createRule(long filterId, MailFilterRule rule)	void	Create a rule for an email filter.

 Mail account filter parameters:

- Field:** Sets the mail field that will be evaluated by the rule.
 Available options are: *From, To, Subject and Content*
 - MailFilterRule.FIELD_FROM
 - MailFilterRule.FIELD_TO
 - MailFilterRule.FIELD_SUBJECT
 - MailFilterRule.FIELD_CONTENT
 - MailFilterRule.FIELD_ATTACHMENT
- Operation:** Set the operation that will be used for the evaluation process.
 Available options are: *Contains and Equal*
 - 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 whether the rule is enabled or not.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailFilterRule rule = new MailFilterRule();
                rule.operation = MailFilterRule.OPERATION_CONTAINS;
                rule.value = "test";
                rule.field = MailFilterRule.FIELD_FROM;
                rule.active = false;
                long filterId = 1; // Valid filter id
                ws.mail.createRule(filterId, rule);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

updateRule

Description:

Method	Return values	Description
updateRule(MailFilterRule rule)	void	Update a rule of an email account.



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 used 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 whether the rule is enabled or not.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
```

```

    {
        ws.login(user, password);
        List<MailFilterRule> rules = ws.getMailFilterRules(filter.id);
        if (rules != null && rules.Count > 0)
        {
            // Update rule
            MailFilterRule rule = new MailFilterRule();
            rule.id = rules[0].id;
            rule.operation = MailFilterRule.OPERATION_EQUALS;
            rule.value = "Any";
            rule.field = MailFilterRule.FIELD_SUBJECT;
            rule.active = true;
            ws.mail.updateRule(rule);
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

deleteRule

Description:

Method	Return values	Description
deleteRule(long ruleId)	void	Delete a rule of an email account.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long ruleId = 1; // Valid rule id
                ws.mail.deleteRule(ruleId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}  
}
```

getFilterRules

Description:

Method	Return values	Description
getFilterRules(long filterId)	List<MailFilterRule>	Retrieve a list of all associated filter rules.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long filterId = 1; // Valid filter id  
                foreach (MailFilterRule mfrule in ws.mail.getFilterRules(filterId))  
                {  
                    System.Console.WriteLine(mfrule.toString());  
                }  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

forwardEmail

Description:

Method	Return values	Description

forwardEmail(String uuid, List<String> users, List<String> roles, List<String> mails, String message)	void	Forward an email to a list of users, roles, or external email addresses.
--	-------------	--

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                string mailUuid = "f123a950-0329-4d62-8328-0ff500fd42db";
                List<string> users = new List<string>();
                users.Add("userTest");
                List<string> roles = new List<string>();
                roles.Add("ROLE_USER");
                List<string> mails = new List<string>();
                mails.Add("test@none.com");
                ws.mail.forwardEmail(mailUuid, users, roles, mails, "Any message");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getPdf

Description:

Method	Return values	Description
getPdf(String uuid)	Stream	Returns a PDF of the email.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Mail mail = ws.mail.getMailProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
                Stream stream = ws.mail.getPdf(mail.uuid);
                FileStream mailFile = new FileStream("D:\\\" + mail.subject + ".pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
                stream.CopyTo(mailFile);
                stream.Close();
                mailFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

saveAsPdf

Description:

Method	Return values	Description
saveAsPdf(String uuid, String newName, bool propertyGroups, bool security)	Document	Save the email as a PDF in the OpenKM repository.

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

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



Additional:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Mail mail = ws.mail.getMailProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
                Document docPdf = ws.mail.saveAsPdf(mail.uuid, mail.subject + ".pdf", true, true);
                System.Console.WriteLine("Document pdf: " + docPdf.path);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getExtractedText

Description:

Method	Return values	Description
getExtractedText(String uuid)	String	Get the extracted text.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
```

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

try
{
    ws.login(user, password);
    System.Console.WriteLine(ws.mail.getExtractedText("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
```

Node samples

Basics



Example of UUID:

- Using UUID -> "373bcdd0-c082-4e7b-addd-e10ef813946e";

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Node methods from the "**node**" class as shown below:

```
ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb")
```

Methods

getVersionHistory

Description:

Method	Return values	Description
getVersionHistory(String uuid)	List<Version>	Returns a list of the version history of a document.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Version> versions = ws.node.getVersionHistory("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

restoreVersion

Description:

Method	Return values	Description
restoreVersion(String uuid, String versionName)	void	Restore a document to a specific version.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.restoreVersion("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7", "1.0");
            }
        }
    }
}

```

```
    }  
    catch (Exception e)  
    {  
        System.Console.WriteLine(e.ToString());  
    }  
}  
}
```

deleteVersion

Description:

Method	Return values	Description
deleteVersion(String uuid, String versionName)	void	Delete a specific version.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
using com.openkm.sdk4csharp.impl;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                ws.node.deleteVersion("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7", "1.0");  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

purgeVersionHistory

Description:

Method	Return values	Description

purgeVersionHistory(String uuid)	void	Purge the version history of a document.
---	-------------	--

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.purgeVersionHistory("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

mayBePromotedAsRecord

Description:

Method	Return values	Description
mayBePromotedAsRecord(String uuid, bool fullEvaluation)	PromoteAsRecordEvaluation	Returns a PromoteAsRecordEvaluation object.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.mayBePromotedAsRecordNode("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7", false);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

promoteAsRecord

Description:

Method	Return values	Description
promoteAsRecord(String uuid)	void	Promote as a record.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.promoteAsRecord("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}
```

isElectronicRecordPath

Description:

Method	Return values	Description
isElectronicRecordPath(String uuid)	Boolean	Returns true if the node is an electronic record.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                bool isERP = ws.node.isElectronicRecordPath("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

degradeRecord

Description:

Method	Return values	Description
degradeRecord(String uuid)	void	Degrade a record.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.degradeRecord("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getElectronicRecordInPath

Description:

Method	Return values	Description
getElectronicRecordInPath(String uuid)	Record	Get the electronic record in the path.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```
        try
        {
            ws.login(user, password);
            Record record = ws.node.getElectronicRecordInPath("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

subscribe

Description:

Method	Return values	Description
subscribe(String uuid)	void	Adds a subscription to a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.subscribe("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

unsubscribe

Description:

Method	Return values	Description
unsubscribe(String uuid)	void	Deletes a subscription from a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.unsubscribe("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

importZip

Description:

Method	Return values	Description
importZip(String uuid, FileStream content)	void	Import a zip file.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{

```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            String folderId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7";
            FileStream zipFile = new FileStream("D:\\Test.zip", FileMode.Open, FileAccess.Read);
            ws.node.importZip(folderId, zipFile);
            zipFile.Close();
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

exportZip

Description:

Method	Return values	Description
exportZip(List<String> uuids, bool withPath, bool background)	Stream	Export as a zip file.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<string> uuids = new List<string>();
                uuids.Add("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
                uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
                // Get stream exportZip
            }
        }
    }
}

```

```

        Stream s = ws.node.exportZip(uuids, true, true);
        // Save as zip file
        FileStream zipFile = new FileStream("D:\\Testing\\Test.zip", FileMode.OpenOrCreate, FileAccess.ReadWrite);
        s.CopyTo(zipFile);
        zipFile.Close();
        s.Close();
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

unZip

Description:

Method	Return values	Description
unZip(String uuid, String dstId)	void	Unzip a file.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String zipFileId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                String folderId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7";
                ws.node.unZip(zipFileId, folderId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getNodeByUuid

Description:

Method	Return values	Description
getNodeByUuid(String uuid)	Node	Get a node by UUID.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Node node = ws.node.getNodeByUuid("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
                System.Console.WriteLine(node.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getChildrenNodesPaginated

Description:

Method	Return values	Description
getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes)	SimpleNodeBaseList	Get paginated child nodes.

 The parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

- The parameter "offset" skips that many results before beginning to return results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> filteredTypes = new List<int>();
                filteredTypes.Add(1);
                String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                ChildNodeList nodeList = ws.node.getChildrenNodesPaginated(folderId, 0, 10, "", NodesPaginationInfo.O

            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getChildrenNodesPaginated

Description:

Method	Return values	Description
getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes, String pluginName)	SimpleNodeBaseList	Get children nodes paginated.



The parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> filteredTypes = new List<int>();
                filteredTypes.Add(1);
            }
        }
    }
}
```

```

        String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
        ChildNodeList nodeList = ws.node.getChildrenNodesPaginated(folderId, 0, 10, "", NodesPaginationInfo.O
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getChildrenNodesByCategoryPaginated

Description:

Method	Return values	Description
getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes)	SimpleNodeBaseList	Get children nodes by category paginated.
 The parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query. - The parameter "limit" is used to limit the number of results returned. - The parameter "offset" skips that many results before beginning to return results.  For example, if your query has 1000 results, but you only want to return the first 10, you should use these values: - limit=10 - offset=0 Now suppose you want to show the results from 11-20, you should use these values: - limit=10 - offset=10		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> filteredTypes = new List<int>();
                filteredTypes.Add(1);
                String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                ChildNodeList nodeList = ws.node.getChildrenNodesByCategoryPaginated(folderId, 0, 10, "", NodesPagir
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getChildrenNodesByCategoryPaginated

Description:

Method	Return values	Description
getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes, String pluginName)	SimpleNodeBaseList	Get children nodes by category paginated.
 The parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query. The parameter "limit" is used to limit the number of		

results returned.

- The parameter "offset" skips that many results before beginning to return results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> filteredTypes = new List<int>();
                filteredTypes.Add(1);
                String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                ChildNodeList nodeList = ws.node.getChildrenNodesByCategoryPaginated(folderId, 0, 10, "", NodesPagir
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getBreadcrumb

Description:

Method	Return values	Description
getBreadcrumb(String uuid)	List<BreadcrumbItem>	Get the breadcrumb.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                List<BreadcrumbItem> list = ws.node.getBreadcrumb(folderId);

            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getNodesFiltered

Description:

Method	Return values	Description
getNodesFiltered(List<String> uuids)	List<Node>	Returns a node list.

Example:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<string> uuids = new List<string>();
                uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
                List<Node> nodes = ws.node.getNodesFiltered(uuids);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

evaluateDownloadZip

Description:

Method	Return values	Description
evaluateDownloadZip(List<String> uuids)	ZipDownloadEvaluationResult	Returns a ZipDownloadEvaluationResult object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            List<string> uuids = new List<string>();
            uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
            ZipDownloadEvaluationResult zdeResult = ws.node.evaluateDownloadZip(uuids);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

generateDownloadToken

Description:

Method	Return values	Description
generateDownloadToken(String uuid, bool preview, bool oneTimeUse, DateTime? validUntil)	String	Generates a node download link.

i When the parameter preview is set to true, the token will be generated for preview purposes.

The preview token expires by default after one minute.

The user must build the download URL with the returned token:

- Download: <http://localhost:8080/openkm/Download?DTK=token>
- Preview: <http://localhost:8080/openkm/Download?PTK=token>

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
        }
    }
}

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            DateTime? validUntil = new DateTime(2023, 2, 10);
            System.Console.WriteLine(ws.node.generateDownloadToken("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7",
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

restore

Description:

Method	Return values	Description
restore(String uuid)	Node	Restore a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Node node = ws.node.restore("0f6463f3-4d36-4091-b518-4fe7c353ee70");
                System.Console.WriteLine(node.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

hasNodesLockedByOtherUser

Description:

Method	Return values	Description
hasNodesLockedByOtherUser(String uuid)	bool	Indicates if the node is blocked by other users.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                if(ws.node.hasNodesLockedByOtherUser("1ec49da9-1746-4875-ae32-9281d7303a62"))
                {
                    System.Console.WriteLine("Is blocked");
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setComment

Description:

Method	Return values	Description
setComment(String uuid, String versionName, String comment)	void	Sets the comment for a specific node version.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.setComment("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43", "1.5", "Update comment");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getPaginatorConfig

Description:

Method	Return values	Description
getPaginatorConfig(String pluginName)	NodePaginatorConfig	Returns a NodePaginatorConfig object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            NodePaginatorConfig nodePaginatorConfig = ws.node.getPaginatorConfig("com.openkm.plugin.paginate");
            Console.WriteLine(nodePaginatorConfig.toString());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getPaginatorPlugins

Description:

Method	Return values	Description
getPaginatorPlugins()	PaginatorPluginList	Returns a PaginatorPluginList object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                PaginatorPluginList list = ws.node.getPaginatorPlugins();
                foreach (PaginatorPlugin paginatorPlugin in list.paginatorPlugins)
                {
                    Console.WriteLine("Name: " + paginatorPlugin.name);
                    Console.WriteLine("ObjClass: " + paginatorPlugin.objClass);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
}
```

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 document is allowed to unlock.

A locked document cannot be modified by other users.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.lock("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

forceLock

Description:

Method	Return values	Description
forceLock(String uuid)	LockInfo	Force-locks a node and returns an object with the lock information.



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

A locked document cannot be modified by other users.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.forceLock("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

unlock

Description:

Method	Return values	Description
unlock(String uuid)	LockInfo	Force-unlocks a locked node.



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

A locked document cannot be modified by other users.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.unlock("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

forceUnlock

Description:

Method	Return values	Description
forceUnlock(String uuid)	void	Unlocks a locked node.
This method allows unlocking any locked document. It is not necessary that this action be executed by the same user who previously performed the checkout lock action.  This action can only be done by a superuser (user with ROLE_ADMIN).		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{

```

```
public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            ws.node.forceUnlock("f123a950-0329-4d62-8328-0ff500fd42db");
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

Note samples

Basics

In most methods, you'll see the parameter named "**nodeId**". 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.newInstance(host);
```

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

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



Once you are logged in with the web services, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Note methods from "**note**" class as shown below:

```
ws.note.add("373bcdd0-c082-4e7b-addd-e10ef813946e", "the note text");
```

Methods

add

Description:

Method	Return values	Description
add(String nodeId, String text)	Note	Adds a note to a node and returns a Note object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.note.add("f123a950-0329-4d62-8328-0ff500fd42db", "the note text");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

get

Description:

Method	Return values	Description
get(String noteId)	Note	Retrieves the note.

 The noteId is a UUID.
 The Node object has a variable named path; in that case, the path contains a UUID.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)

```

```

    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            List<Note> notes = ws.note.list("f123a950-0329-4d62-8328-0ff500fd42db");
            if (notes.Count > 0)
            {
                System.Console.WriteLine(ws.note.get(notes[0].path));
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

delete

Description:

Method	Return values	Description
delete(String noteId)	Note	Deletes a note.

 The noteId is a UUID.
 The Node object has a variable named path; in that case, the path contains a UUID.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {

```

```

        ws.login(user, password);
        List<Note> notes = ws.note.list("f123a950-0329-4d62-8328-0ff500fd42db");
        if (notes.Count > 0)
        {
            ws.note.delete(notes[0].path);
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

set

Description:

Method	Return values	Description
set(String noteId, String text)	void	Changes the note text.
 The noteId is a UUID. The Node object has a variable named path; in that case, the path contains a UUID.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Note> notes = ws.note.list("f123a950-0329-4d62-8328-0ff500fd42db");
                if (notes.Count > 0)
                {
                    ws.note.set(notes[0].path, "text modified");
                }
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```

        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

list

Description:

Method	Return values	Description
list(String nodeId)	List<Note>	Retrieves a list of all notes for a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Note> notes = ws.note.list("f123a950-0329-4d62-8328-0ff500fd42db");
                foreach (Note note in notes)
                {
                    System.Console.WriteLine(note);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getNotesHistories

Description:

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

getNotesHistories(String nodeId)	List<NoteHistory>	Retrieves a list of all note histories of a node.
---	--------------------------------	---

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<NoteHistory> notes = ws.note.getNotesHistory("f123a950-0329-4d62-8328-0ff500fd42db");
                foreach (NoteHistory note in notes)
                {
                    System.Console.WriteLine(note);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Notification samples

Basics

Suggested code sample

First, you must create the web service object:

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

Then, you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web service, the session is kept in the web service object. Then you can use the other API methods.

At this point you can use all the Notification methods from the "**notification**" class as 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, bool attachment)	void	Sends a mail notification.
 The parameter uuids contains the UUIDs of the nodes (document, folder, mail, or record). The parameter users contains a set of OpenKM users to be notified. The parameter roles contains a set of OpenKM roles to be notified. The parameter mails contains a set of email addresses - usually external email addresses - to be notified. The parameter message is the body of the email. When the attachment value is true, the node is attached to the email.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<String> uuids = new List<string>();
                uuids.Add(doc.uuid);
                List<String> users = new List<string>();
                users.Add("jperez");
                List<String> roles = new List<string>();
                roles.Add("ROLE_USER");
                List<String> mails = new List<string>();
                mails.Add("test@none.com");
                String message = "Any message"
                ws.notification.notify(uuids, users, roles, mails, message, false);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

PDF samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in to the web service, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the PDF methods from the "**pdf**" class as shown below:

```
Stream stream = ws.pdf.getImage("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", 1);
```

Methods

getImage

Description:

Method	Return values	Description
getImage(String uuid, int page, String size)	Stream	Returns the image of a page.
 The document must be a PDF or convertible to PDF. The value of the uuid is the UUID of the document. The page is the number of the page in the document, starting at 1. The size value is optional. When set, it is used to resize the image of the page; otherwise, the default value is used.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream stream = ws.pdf.getImage("1ec49da9-1746-4875-ae32-9281d7303a62", 1, null);
                FileStream destFile = new FileStream("D:\\test.png", FileMode.OpenOrCreate, FileAccess.ReadWrite);
                stream.CopyTo(destFile);
                destFile.Close();
                stream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

split

Description:

Method	Return values	Description
split(String uuid, String dstId, String baseName, List<int> pages)	List<Document>	Returns the split documents.



The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of **baseName** is the base name to create a split document, for example, if baseName value is "test" and pages value is "1,3", the files created will be "test-001.pdf" and "test-003.pdf".

The value of **pages** is the number of pages to be split.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> pages = new List<int>();
                pages.Add(5);
                pages.Add(10);
                List<Document> docs = ws.pdf.split("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "1ec49da9-1746-4875-a");
                foreach(Document doc in docs)
                {
                    System.Console.WriteLine(doc.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

extract

Description:

Method	Return values	Description
extract(String uuid, String dstId, String name, List<int> pages)	Document	Returns a new document with the extracted pages.
 The document must be a PDF or convertible to PDF.		
The value of the uuid is the UUID of the document.		
The value of the dstId is the UUID of the destination (folder or record).		

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be extracted.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> pages = new List<int>();
                pages.Add(5);
                pages.Add(10);
                Document result = ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "1ec49da9-1746-4875-ae3");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

remove

Description:

Method	Return values	Description
remove(String uuid, String dstId, String name, List<int> pages)	Document	Returns the document with the pages removed.
 The document must be a PDF or convertible to PDF.		
The value of the uuid is the UUID of the document.		

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be removed.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> pages = new List<int>();
                pages.Add(5);
                pages.Add(10);
                Document doc = ws.pdf.remove("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "1ec49da9-1746-4875-ae32");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

rotate

Description:

Method	Return values	Description
rotate(String uuid, String dstId, String name, int angle, List<int> pages)	Document	Returns the document with the pages rotated.
 The document must be a PDF or convertible to PDF.		

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **angle** is the rotation applied to the chosen pages.

The value of **pages** is the number of pages to be rotated.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> pages = new List<int>();
                pages.Add(5);
                pages.Add(10);
                Document doc = ws.pdf.rotate("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "1ec49da9-1746-4875-ae32-9");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

insertPages

Description:

Method	Return values	Description
insertPages(String uuid, InsertPagesRequest request)	Document	Returns the new document created with the inserted pages.



The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the target document where the pages will be inserted.

The value of **InsertPagesRequest** is an object that has a List of **PageInsertOperation** named **insertOperations**, a String **dstId**, that contains the UUID of the destination folder, and a String **name** for the newly generated document.

PageInsertOperation object has a String **srcId**, the UUID of the source file of the pages, a List of **Integer** named **pages** that stores the positions of the pages in the source PDF, and an int **insertFromPage**, the position where the pages have to be located in the target PDF.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);

                // Create the insert pages request
                InsertPagesRequest request = new InsertPagesRequest();
                request.name = "merged_document.pdf"; // Name for the new document
                request.dstId = fld.uuid;

                // Create a page insertion operation
                PageInsertOperation operation = new PageInsertOperation();
                operation.srcId = srcDoc.uuid; // Source document UUID
                operation.pages = new List<int> { 1, 2, 3 }; // Pages to insert from source
                operation.insertFromPage = 1; // Insert at position 1 in target document

                request.insertOperations.Add(operation);

                // Call insertPages with target document UUID and request
                Document result = ws.pdf.insertPages(srcDoc.uuid, request);

                System.Console.WriteLine("New document created: " + result.uuid);
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```
        System.Console.WriteLine(e.ToString());  
    }  
}
```

optimize

Description:

Method	Return values	Description
optimize(String uuid)	bool	Returns true when optimized successfully.

 The UUID of the node must be a PDF document.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password); ;  
                bool result = ws.pdf.optimize("e7d6860a-7e0b-4823-bd3d-75f88be1eedf");  
                System.Console.WriteLine("Optimize success: " + result);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

Plugin samples

Basics

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web services, the session is kept in the web service object. Then you can use the other API methods.

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

```
ws.plugin.executePluginPostReturnFile("com.openkm.plugin.rest.TestGetDocumentRestPlugin", parameters, null);
```

Methods

executePluginPost

Description:

Method	Return values	Description
executePluginPost(String className, Dictionary<String, String> parameters, Type clazz, FileStream fs)	Object	Returns the object value resulting from execution of a class that implements RestPlugin.



- The "className" value must be the canonical class name of a class that implements the RestPlugin interface.
- The "parameters" map contains values that will be used by the class specified in className.
- The "clazz" value should be the Class of the returned object. It's used by REST to unmarshal the result of the query.

- The "fs" file stream allows uploading a document.



This method executes a REST call using POST.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream(@"D:\Testing\plugging.docx", FileMode.Open, FileAccess.Read);
                Dictionary<String, String> parameters = new Dictionary<String, String>();
                parameters.Add("param1", "value1");
                parameters.Add("param2", "value2");
                String res = (String) ws.plugin.executePluginPost("com.openkm.plugin.rest.TestRestPlugin", parameters, fs);
                fs.Close();
                System.Console.WriteLine(res);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executePluginPostReturnFile

Description:

Method	Return values	Description
executePluginPostReturnFile(String className, Dictionary<String, String> parameters, FileStream fs)	Stream	Returns an InputStream resulting from execution of a class that implements RestPlugin.



- The "className" value must be the canonical class name of a class that implements the RestPlugin interface.
- The "parameters" map contains values that will be used by the class specified in className.
- The "is" stream allows uploading a document.



This method executes a REST call using POST.

The RestPlugin must return a Response object. Take a look at the following sample implementation of the RestPlugin:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

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

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
```

```

        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(isContent);
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
    }
}

```

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream(@"D:\Testing\plugging.docx", FileMode.Open, FileAccess.Read);
                Dictionary<String, String> parameters = new Dictionary<String, String>();
                parameters.Add("docId", "/okm:root/test.pdf");
                Stream tmpFile = ws.plugin.executePluginPostReturnFile("com.openkm.plugin.rest.TestGetDocumentRes
                FileStream destFile = new FileStream(@"D:\Testing\plugging.pdf", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
                fs.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

executePluginGet

Description:

Method	Re
--------	----

executePluginGet(String className, Dictionary<String, String> parameters, Type clazz)

Obj



- The "className" value must be the canonical class name of a class that implements the RestPlugin interface.
- The "parameters" map contains values that will be used by the class specified in className.
- The "clazz" value should be the Class of the returned object. It's used by REST to unmarshal the result of the call.



This method executes a REST call using GET.

You can also execute it directly from the browser; take the following URL as a sample:

`http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?className=com.openkm.plugin.`

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4sharp;
using com.openkm.sdk4sharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> parameters = new Dictionary<String, String>();
                parameters.Add("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
                String res = (String) ws.plugin.executePluginGet("com.openkm.plugin.rest.TestRestPlugin", parameters, typeof(String));
                System.Console.WriteLine(res);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executePluginGetReturnFile

Description:

Method

executePluginGetReturnFile(String className, Dictionary<String, String> parameters)

- The "className" value must be the canonical class name of a class that implements the RestPlugin interface
- The "parameters" map contains values that will be used by the class specified in className.



This method executes a REST call using GET.

The RestPlugin must return a Response object. Take a look at the following sample implementation of the RestPlug

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

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

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
    }
}
```

```

        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(isContent);
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
    }
}

```

You can also execute it directly from the browser; use the following URL as a sample:

<http://localhost:8080/openkm/services/rest/plugin/executeGetPluginReturnFile?className=com.openkm.plugin.rest.TestGetDocumentRestPlugin>

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> parameters = new Dictionary<String, String>();
                parameters.Add("docId", "/okm:root/test.pdf");
                Stream tmpFile = ws.plugin.executePluginGetReturnFile("com.openkm.plugin.rest.TestGetDocumentRestPlugin");
                FileStream destFile = new FileStream(@"D:\Testing\pluging.pdf", FileMode.OpenOrCreate);
                tmpFile.CopyTo(destFile);
                destFile.Close();
                tmpFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
```

Property samples

Basics

In almost all methods, you'll see the parameter named "**nodeId**". 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.newInstance(host);
```

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

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



Once you are logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

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

```
ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-bd78-e26d7c6fb648");
```

Methods

addCategory

Description:

Method	Return values	Description
addCategory(String nodeId, String catId)	void	Sets a relation between a category and a node.
The value of the catId parameter should be a category folder UUID or path.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.addCategory("f123a950-0329-4d62-8328-0ff500fd42db", "055e4384-7c70-4456-b32b-f5a55a");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

removeCategory

Description:

Method	Return values	Description
removeCategory(String nodeId, String catId)	void	Removes a relation between a category and a node.
The value of the catId parameter should be a category folder UUID or path.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            ws.property.removeCategory("f123a950-0329-4d62-8328-0ff500fd42db", "ac165cab-a62a-4b17-89fc-2480");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

addKeyword

Description:

Method	Return values	Description
addKeyword(String nodeId, String keyword)	void	Adds a keyword to a node.

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two_words" (the character "_" is used as the separator).

 Also, we suggest you add a keyword in lowercase because OpenKM is case sensitive.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.addKeyword("f123a950-0329-4d62-8328-0ff500fd42db", "test");
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

removeKeyword

Description:

Method	Return values	Description
removeKeyword(String nodeId, String keyword)	void	Removes a keyword from a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.removeKeyword("f123a950-0329-4d62-8328-0ff500fd42db", "test");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setEncryption

Description:

Method	Return values	Description

setEncryption(String nodeId, String cipherName)	void	Marks a document as encrypted binary data in the repository.
--	-------------	--

The parameter nodeId should be a document node.

The parameter cipherName stores information about the encryption mechanism.



This method does not perform any kind of encryption; it simply marks in the database that a document is encrypted.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.setEncryption("f123a950-0329-4d62-8328-0ff500fd42db", "phrase");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

unsetEncryption

Description:

Method	Return values	Description
unsetEncryption(String nodeId)	void	Marks a document as normal binary data in the repository.
The parameter nodeId should be a document node.		



This method does not perform any kind of decryption; it simply marks in the database that a document has been decrypted.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.unsetEncryption("f123a950-0329-4d62-8328-0ff500fd42db");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setSigned

Description:

Method	Return values	Description
setSigned(String nodeId, boolean signed)	void	Marks a document as signed or unsigned in the repository.
The parameter nodeId should be a document node.		
 This method does not perform any kind of digital signature process; it simply marks in the database that a document is signed.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.setSigned("f123a950-0329-4d62-8328-0ff500fd42db", true);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

isSigned

Description:

Method	Return values	Description
isSigned(String uuid)	bool	Returns a boolean indicating whether the document is signed.
The parameter uuid should be a document node.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    System.Console.WriteLine("Is the document signed: " + ws.property.isSigned("6330d2a0-529f-4c14-baa1
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
```

PropertyGroup samples

Basics



In an older OpenKM version, we called "**Metadata Groups**" "**Property Groups**".

Although we understand this name does not help much in identifying these methods as metadata ones, for historical reasons we continue to maintain the nomenclature.

For more information about [Metadata](#).

In almost all methods, you'll see the parameter named "**nodeId**". The value of this parameter can be the UUID of a valid document, folder, mail, or record.



Example of a nodeId:

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



The class `com.openkm.sdk4j.util.ISO8601` should be used to set and parse metadata date fields. The values of metadata fields of type date are stored in the application in ISO-8601 basic format.

To convert a retrieved metadata field of type date to a valid date, use:

```
DateTime date = ISO8601.parseBasic(metadataFieldValue);
```

To save a date value into a metadata field of type date, use:

```
DateTime date = DateTime.now; // Present date
String metadataFieldValue = ISO8601.formatBasic(date);
// metadataFieldValue can be saved into repository metadata field of type date
```

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged in to the web service, the session is kept in the webservice object. You can then use the

 other API methods.

At this point you can use all the Property Group methods from the "**propertyGroup**" class as shown below:

```
ws.propertyGroup.addGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", propertiesMap);
```

Methods

addGroup

Description:

Method	Return values	Description
addGroup(String nodeId, String grpName, Dictionary<String, String> propertiesMap)	void	Adds a metadata group to a node.

The grpName should be a valid Metadata group name.

 It is not mandatory to set all field values in the propertiesMap parameter; it is enough to include the fields whose values you wish to change.

 The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
    "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

 To add several values to a metadata field of type **multiple** and to add a value to a metadata field of type **simple**, like this:

```
<select label="Multiple" name="okp:consulting.multiple" type="multiple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>

<select label="Simple" name="okp:consulting.simple" type="simple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
```

```
<option label="Three" value="three" />
</select>
```

You should do:

```
properties.Add("okp:consulting.multiple", "[one,two]");
properties.Add("okp:consulting.simple", "[one]");
```

Where **"one"** and **"two"** are valid values, the character **" , "** is used as a separator, and the options should go between **"["** and **"]"**.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<string, string> props = new Dictionary<string, string>();
                DateTime date = DateTime.Now;
                props.Add("okp:consulting.name", "test");
                props.Add("okp:consulting.date", ISO8601.formatBasic(date));
                props.Add("okp:consulting.important", "true");
                props.Add("okp:consulting.comment", "test");
                ws.propertyGroup.addGroup("f0064cf3-4776-44c2-868a-f08697a6957e", "okg:consulting", props);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

removeGroup

Description:

Method	Return values	Description

removeGroup(String nodeId, String grpName)	void	Removes a metadata group from a node.
The grpName should be a valid Metadata group name.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.propertyGroup.removeGroup("f123a950-0329-4d62-8328-0ff500fd42db", "okg:consulting");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getGroups

Description:

Method	Return values	Description
getGroups(String nodeId)	List<PropertyGroup>	Retrieves a list of metadata groups assigned to a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (PropertyGroup pGroup in ws.propertyGroup.getGroups("f123a950-0329-4d62-8328-0ff500fd42d1"))
                {
                    System.Console.WriteLine(pGroup.name);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getAllGroups

Description:

Method	Return values	Description
getAllGroups()	List<PropertyGroup>	Retrieves a list of all metadata groups configured in the application.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (PropertyGroup pGroup in ws.propertyGroup.getAllGroups())
                {
                    System.Console.WriteLine(pGroup.name);
                }
            }
        }
    }
}

```

```

    }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getAllGroups

Description:

Method	Return values	Description
getAllGroups(String uuid)	List<PropertyGroup>	Retrieves a list of all metadata groups defined in the application.
 This method allows, through the "Get all metadata groups" automation task event, filtering the list of groups allowed in a specific node.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (PropertyGroup pGroup in ws.propertyGroup.getAllGroups("8cd1e072-8595-4dd3-b121-41d622c4"))
                {
                    System.Console.WriteLine(pGroup.name);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
}
```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm(String uuid, String grpName)	List<FormElement>	Retrieves a list of all metadata group element definitions.

The grpName should be a valid Metadata group name.



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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupForm("f123a950-0329-4d62-8328-8328-8328-8328-8328-8328"))
                {
                    System.Console.WriteLine(fElement);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getPropertyGroupFormDefinition

Description:

Method	Return values	Description
getPropertyGroupFormDefinition(String grpName)	List<FormElement>	Retrieves a list of all metadata group element definitions.
The grpName should be a valid Metadata group name.		
 The method is usually used to display empty form elements for creating new metadata values.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupFormDefinition("okg:consulting"))
                {
                    System.Console.WriteLine(fElement);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getPropertyGroupFormDefinition

Description:

Method	Return values	Description
getPropertyGroupFormDefinition(String uuid, String grpName)	List<FormElement>	Retrieves a list of all metadata group element definitions.
The grpName should be a valid Metadata group name.		
 The method is usually used to display empty form elements for creating new metadata values.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupFormDefinition("afe29d6e-7769-4c
                {
                    System.Console.WriteLine(fElement);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setProperties

Description:

Method

setProperties(String nodeId, String grpName, Dictionary<String, String> properties)

The grpName should be a valid Metadata group name.



Before changing metadata, you **should have the group added to the node** (see addGroup method); otherwise an e



It is not mandatory to set all field values in the properties parameter; it is enough to include the fields whose values



The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//openkm//DTD Property Groups 2.0//EN"
"http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```



To add several values to a metadata field of type **multiple** and to add a value to a metadata field of type **simple** like

```
<select label="Multiple" name="okp:consulting.multiple" type="multiple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>

<select label="Simple" name="okp:consulting.simple" type="simple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>
```

You should do:

```
properties.Add("okp:consulting.multiple", "[one,two]");
properties.Add("okp:consulting.simple", "[one]");
```

Where **"one"** and **"two"** are valid values, the character **" , "** is used as a separator, and the options should go between

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("okp:consulting.name", "new name");

                //The date fields must be saved with basic ISO 8601 format
                properties.Add("okp:consulting.date", ISO8601.formatBasic(DateTime.Now));
                ws.propertyGroup.setProperties("f123a950-0329-4d62-8328-0ff500fd42db", "okg:consulting", properties);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

hasGroup

Description:

Method	Return values	Description
hasGroup(String nodeId, String grpName)	Boolean	Returns a boolean that indicates whether the node has a metadata group.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    System.Console.WriteLine("Have metadata group:" + ws.propertyGroup.hasGroup("f123a950-0329-4d62-4030-b10123456789"));
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

getRegisteredDefinition

Description:

Method	Return values	Description
getRegisteredDefinition()	String	Returns the XML metadata groups definition.



The method can only be executed by a superuser (ROLE_ADMIN user).

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.propertyGroup.getRegisteredDefinition());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}  
}
```

registerDefinition

Description:

Method	Return values	Description
registerDefinition(InputStream is, String pgName)	void	Sets the XML metadata groups definition into the repository.

 The method can only be executed by a superuser (ROLE_ADMIN user).

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using System.IO;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8180/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                FileStream fs = new FileStream("E:\\PropertyGroups.xml", FileMode.Open);  
                ws.propertyGroup.registerDefinition(fs, "okg:testing");  
                fs.Dispose();  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getSuggestions

Description:

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

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

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

 The sample below is based on this Metadata group definition:


```

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

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    foreach(String value in ws.propertyGroup.getSuggestions("f123a950-0329-4d62-8328-0ff500fd42db","okg
    {
        System.Console.WriteLine(value);
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

getProperties

Description:

Method	Return values	Description
getProperties(String nodeId, String grpName)	Dictionary<String, String>	Retrieves a dictionary of (key, value) pairs with metadata group values for a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties = ws.propertyGroup.getProperties("f123a950-0329-4d62-8328-0ff500fd42db","okg:technology");

                foreach (KeyValuePair<String, String> pair in properties)
                {
                    System.Console.WriteLine(pair.Key + "-> " + pair.Value);
                }
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getPropertiesByVersion

Description:

Method	Return values	Description
getPropertiesByVersion(String nodeId, String grpName,String versionName)	Dictionary<String, String>	Retrieves a dictionary of (key, value) pairs with metadata group values for a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties = ws.propertyGroup.getPropertiesByVersion("f123a950-0329-4d62-8328-0ff500fd42db","okg:te

                foreach (KeyValuePair<String, String> pair in properties)
                {
                    System.Console.WriteLine(pair.Key + "-> " + pair.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getGroupsByVersion

Description:

Method	Return values	Description
getGroupsByVersion(String nodeId, String versionName)	List<PropertyGroup>	Retrieves a list of metadata groups assigned to a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (PropertyGroup pGroup in ws.propertyGroup.getGroupsByVersion("f123a950-0329-4d62-8328-0f
                {
                    System.Console.WriteLine(pGroup.name, "1.1");
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getGroup

Description:

Method	Return values	Description
getGroup(String grpName)	PropertyGroup	Gets the property group definition.

Example:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                PropertyGroup pg = ws.propertyGroup.getGroup("okg:consulting");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getSuggestBoxKeyValue

Description:

Method	Return 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"</pre>		

```

        valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
    </property-group>
</property-groups>

```

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String grpName = "okg:consulting";
                String propertyName = "okp:consulting.suggestbox";
                String key = "es";
                String result = ws.propertyGroup.getSuggestBoxKeyValue(grpName, propertyName, key);
                Console.WriteLine("Value: " + result);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getSuggestBoxKeyValuesFiltered

Description:

Method	Return values	Description
getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)	Dictionary<String, String>	Retrieves a dictionary (key, value) pair 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String grpName = "okg:consulting";
                String propertyName = "okp:consulting.suggestbox";
                String filter = "Port";
                Dictionary<String, String> result = ws.propertyGroup.getSuggestBoxKeyValuesFiltered(grpName, propertyName, filter);
                foreach (var item in result)
                {
                    Console.WriteLine(item.key + ":" + item.value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

validateField

Method	Return values	
validateField(String value, String className, List<String> uuids)	String	Return



- The "className" value must be the canonical class name of a class that implements the FieldValidator interface.



Example of the Form validator implementation.

More information at [Creating your own Form Validator plugin](#).

In this example, two identical comments will not be allowed in the metadata field named okp:technology.comment.

```
package com.openkm.plugin.form.validator;

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

import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.form.FieldValidator;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.db.service.LegacySrv;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements FieldValidator {

    @Autowired
    private LegacySrv legacySrv;

    @Autowired
    private OKMRelation okmRelation;

    @Override
    public String getName() {
        return "Duplicated document number";
    }

    @Override
    public String validate(String value, List<String> uuids) {
        String validate = "";

        String token = DbSessionManager.getInstance().getSystemToken();
        String sql = "SELECT RGT_UUID FROM OKM_PGRP_CUR_TECHNOLOGY WHERE RGT_PROPR";
        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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<String> uuids = new List<string>();
                uuids.Add("28ece92f-9708-4d3f-8033-18dc98b9e7f5");
                uuids.Add("5484b622-7c5f-425a-9451-7611c0caf227");

                String value = "test";
                String className = "com.openkm.plugin.form.validator.DuplicateDocumentNumberValidator";
                String message = ws.propertyGroup.validateField(value, className, uuids);
                Console.WriteLine("Validate: " + message);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```



Record samples

On most methods, you'll see the parameter named "**recId**" and "**fldId**". The value of these parameters can be a valid record and folder **UUID**.



Example of recId:

- Using UUID -> "28bb0c8b-3701-4457-a81a-fef7bc56ff33"

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservice, the session is kept in the webservice object. Then you can use the other API methods

At this point you can use all the Record methods from the "**record**" class as shown below:

```
ws.record.create("ada67d44-b081-4b23-bdc1-74181cafb5d", "PKI-100200", "new title", 0)
```

Methods

create

Description:

Method	Return values	Description
create(String fldId, String name, String title, long nodeClass)	Record	Creates a new record and returns a Record object.
Optionally, a title can be set; the other variables of the Record (record) will not have any effect on record creation.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.create(fldId, "PKI-100200", "some title", 0);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getProperties

Description:

Method	Return values	Description
getProperties(String uuid)	Record	Returns the record properties.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    System.Console.WriteLine(ws.record.getProperties("50b7a5b9-89d2-430e-bbc9-6a6e01662a71").path);
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

delete

Description:

Method	Return values	Description
delete(String recId)	void	Deletes a record.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.delete("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

purge

Description:

Method	Return values	Description
purge(String recId)	void	Records are permanently removed from the repository.

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



When a record is purged, it can only be restored from a previous repository backup. The purge action permanently removes the record from the repository.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.purge("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(String uuid, String newName)	void	Renames a record.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.rename("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "PKI-100201");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

move

Description:

Method	Return values	Description
move(String recId, String dstId)	void	Moves a record into a folder or another record.
The values of the dstId parameter should be a folder or record UUID or path.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    ws.record.move("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79861f");
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

copy

Description:

Method	Return values	Description
copy(String recId, String dstId, String newName)	Record	Copies a record into a folder or another record.

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

If the newName parameter 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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.copy("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79861f");
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}

```

isValid

Description:

Method	Return values	Description
isValid(String uuid)	Boolean	Returns a boolean that indicates whether the node is a record.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Is a record:" + ws.record.isValid("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getChildren

Description:

Method	Return values	Description
getChildren(String uuid)	List<Record>	Returns a list of all records whose parent is fldId.

The parameter **uuid** can be a folder or a record node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Record rec in ws.record.getChildren("055e4384-7c70-4456-b32b-f5a55a79861f"))
                {
                    System.Console.WriteLine(rec);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(String uuid, String title)	void	Sets a record title.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            ws.record.setTitle("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "some title");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getPath

Description:

Method	Return values	Description
getPath(String uuid)	String	Converts a record UUID to a record path.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.record.getPath("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
```

setNodeClass

Description:

Method	Return values	Description
setNodeClass(String uuid, long ncId)	void	Sets a record node class.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.setNodeClass("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", 0);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(String uuid, long stage)	void	Sets the disposition stage.

Example:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.setDispositionStage("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", 1);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setDescription

Description:

Method	Return values	Description
setDescription(String uuid, String description)	void	Sets the title.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.setDescription("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "some description");
            }
        }
    }
}

```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

createWizard

Description:

Method	Return values	Description
createWizard(String uuid, String name, String title, long nodeClass)	WizardNode	Creates a record using a wizard.



The WizardNode contains a list of pending actions that should be done to complete the process of record creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                WizardNode wnode = ws.record.createWizard("f123a950-0329-4d62-8328-0ff500fd42db", "REC-001", "Ar
                System.Console.WriteLine(wnode.toString());
            }
        }
    }
}

```

```

        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

createFromTemplate

Description:

Method	Return values	Description
createFromTemplate(String uuid, String dstPath, bool categories, bool keywords, bool notes, bool propertyGroups, bool security Dictionary<String, String> properties)	Record	Creates a new record from the template and returns a Record object.
The uuid parameter is the template UUID of the record. The dstPath can be a folder or a record path. When the template uses metadata groups to fill in fields, these values are mandatory and must be set in the properties parameter. i For more information about Templates and metadata check: Creating templates. i 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 notes parameter is true, the original values of the notes will be copied.		

Example:

```

using System; using System.Collections.Generic;
using System.Linq; using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)

```

```

    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
        try
        {
            ws.login(user, password);
            Dictionary<String, String> properties = new Dictionary<String, String>();
            properties.Add("okp:tpl.name", "Some name");
            DateTime date = DateTime.Now;
            // Value must be converted to String ISO 8601 compliant
            properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
            properties.Add("okp:tpl.language", "[ \"java\" ]");
            Record record = ws.record.createFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "/okm:root");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(String uuid, String dstId, String newName, bool categories, bool keywords, bool propertyGroups, bool notes, bool 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 preserve the same name.



By default, only the binary data and the security grants are copied; the metadata, keywords, etc. of the folder are not copied.

Additional:

- 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:

```
using System; using System.Collections.Generic;
using System.Linq; using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);
            try
            {
                ws.login(user, password);
                ws.record.extendedCopy("ada67d44-b081-4b23-bdc1-74181cafb5d", "8599eab7-ae61-4628-8010-110
                    "new name record", true, true, true, true, true);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createMissingRecords

Description:

Method	Return values	Description
createMissingRecords(String recPath)	String	Create missing records.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
```

```
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    ws.record.createMissingRecords("/okm:root/rec001/rec0011/rec00111");
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
```

Relation samples

Basics

On most methods, you'll see the parameter named "**nodeId**". The value of this parameter can be a valid document, folder, mail or record **UUID** or **path**.



Example of nodeId:

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

Suggested code sample

First, you must create the webservice object:

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

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

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



Once logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

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
getRelationTypes(String type)	List<RelationType>	Retrieves a list of all relations defined for a type.
Available types:		

- `RelationType.BIDIRECTIONAL`
- `RelationType.PARENT_CHILD`
- `RelationType.MANY_TO_MANY`



More information on [Relation types](#).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationType type in ws.relation.getRelationTypes(RelationType.PARENT_CHILD))
                {
                    System.Console.WriteLine(type);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

add

Description:

Method	Return values	Description
add(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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationType type in ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL))
                {
                    // looking for a relation named invoice
                    if (type.title.Equals("invoice"))
                    {
                        ws.relation.add("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79d8e8");
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(long relationId)	void	Deletes a relationship.
 A relation can be deleted only when no node uses it. Otherwise, you'll get an error.		

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationType type in ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL))
                {
                    // looking for a relation named invoice
                    if (type.title.Equals("invoice"))
                    {
                        ws.relation.delete(type.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getRelations

Description:

Method	Return values	Description
getRelations(String uuid)	List<Relation>	Retrieves a list of all relations of a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    foreach (Relation relation in ws.relation.getRelations("055e4384-7c70-4456-b32b-f5a55a79861f"))
    {
        System.Console.WriteLine(relation);
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

getRelationGroups

Description:

Method	Return values	Description
getRelationGroups(String uuid)	List<RelationGroup>	Retrieves a list of all relation groups for a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("055e4384-7c70-4456-b32b-f5a55a79861f"))
                {
                    System.Console.WriteLine(rGroup.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

    }
  }
}

```

addGroup

Description:

Method	Return values	Description
addGroup(String nodeId, String groupName, long type)	void	Adds a relation group to a node.

For a relation group, it only makes sense to apply the relation type `RelationType.MANY_TO_MANY`.


[More information on `Relation types`.](#)

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationType type in ws.relation.getRelationTypes(RelationType.MANY_TO_MANY))
                {
                    if (type.title.Equals("staple"))
                    {
                        ws.relation.addGroup("055e4384-7c70-4456-b32b-f5a55a79861f", "staple group", type.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

addNodeToGroup

Description:

Method	Return values	Description
addNodeToGroup(String nodeId, long groupId)	void	Adds a node to an existing relation group.

A relation group only makes sense with the relation type RelationType.MANY_TO_MANY.


[More information on Relation types.](#)

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("055e4384-7c70-4456-b32b-f5a55a7986"))
                {
                    if (rGroup.name.Equals("staple group"))
                    {
                        ws.relation.addNodeToGroup("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", rGroup.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

deleteGroup

Description:

Method	Return values	Description
deleteGroup(long groupId)	void	Removes a node from a relation group.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662"))
                {
                    if (rGroup.name.Equals("staple group"))
                    {
                        ws.relation.deleteGroup(rGroup.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getGroup

Description:

Method	Return values	Description
getGroup(long groupId)	RelationGroup	Finds a relation group by ID.

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                RelationGroup rg = new RelationGroup();
                foreach (RelationGroup rGroup in ws.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"))
                {
                    if (rGroup.name.Equals("staple group3"))
                    {
                        rg = ws.relation.getGroup(rGroup.id);
                        break;
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setGroupName

Description:

Method	Return values	Description
setGroupName(long groupId, String groupName)	void	Changes the relation group name.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    RelationGroup rg = new RelationGroup();
    foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662
    {
        if (rGroup.name.Equals("staple group3"))
        {
            ws.relation.setGroupName(rGroup.id, "newGroup");
            rg = ws.relation.getGroup(rGroup.id);
            break;
        }
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

getAllRelationGroups

Description:

Method	Return values	Description
getAllRelationGroups(int relationTypeId, String filter, int offset, int limit)	RelationGroupResultSet	Retrieves a list of all relation groups.
 The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. • The parameter "limit" limits the number of results returned. • The parameter "offset" specifies how many results to skip before beginning to return results.		
 For example, if your query has 1000 results, but you only want to return the first 10, you should use these values: • limit=10 • offset=0 Now suppose you want to show the results from 11-20; you should use these values: • limit=10 • offset=10		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int relationTypeId = 1;
                String filter = "";
                RelationGroupResultSet resultSet = ws.relation.getAllRelationGroups(relationTypeId, filter, 0, 10);
                foreach (RelationGroup relationGroup in resultSet.results)
                {
                    Console.WriteLine(relationGroup.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

deleteItemFromGroup

Description:

Method	Return values	Description
deleteItemFromGroup(String nodeId, long groupId)	void	Removes a node from a relation group.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"))
                {
                    if (rGroup.name.Equals("staple group3"))
                    {
                        ws.relation.deleteItemFromGroup("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", rGroup.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Repository samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservice, the session is kept in the webservice object. Then you can use the other API methods.

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

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

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder()	Folder	Returns the Folder object of the node "/okm:root".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
    }
```

```

static void Main(string[] args)
{
    String host = "http://localhost:8080/openkm";
    String username = "okmAdmin";
    String password = "admin";
    OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

    try
    {
        ws.login(user, password);
        System.Console.WriteLine(ws.repository.getRootFolder());
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

getTrashFolder

Description:

Method	Return values	Description
getTrashFolder()	Folder	Returns the Folder object of the node "/okm:trash/{userId}".
The returned folder will be the user's trash folder.  For example, if the method is executed by the "okmAdmin" user, then the returned folder will be "/okm:trash/okmAdmin".		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
            }
        }
    }
}

```

```
        System.Console.WriteLine(ws.repository.getTrashFolder());
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

getTrashFolderBase

Description:

Method	Return values	Description
getTrashFolderBase()	Folder	Returns the Folder object of the node "/okm:trash".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getTrashFolderBase());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getTemplatesFolder

Description:

Method	Return values	Description

getTemplatesFolder()	Folder	Returns the Folder object of the node "/okm:templates".
-----------------------------	---------------	---

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getTemplatesFolder());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getPersonalFolder

Description:

Method	Return values	Description
getPersonalFolder()	Folder	Returns the Folder object of the node "/okm:personal/{userId}".

The returned folder will be the user's personal folder.



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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getPersonalFolder());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getPersonalFolderBase

Description:

Method	Return values	Description
getPersonalFolderBase()	Folder	Returns the Folder object of the node "/okm:personal".

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getPersonalFolderBase());
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```
        {  
            System.Console.WriteLine(e.ToString());  
        }  
    }  
}
```

getMailFolder

Description:

Method	Return values	Description
getMailFolder()	Folder	Returns the Folder object of the node "/okm:mail/{userId}".

The returned folder will be the user's mail folder.



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

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                System.Console.WriteLine(ws.repository.getMailFolder());  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getMailFolderBase

Description:

Method	Return values	Description
getMailFolderBase()	Folder	Returns the Folder object of the node "/okm:mail".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getMailFolderBase());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getCategoriesFolder

Description:

Method	Return values	Description
getCategoriesFolder()	Folder	Returns the Folder object of the node "/okm:categories".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getCategoriesFolder());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

purgeTrash

Description:

Method	Return values	Description
purgeTrash()	void	Permanently removes from the repository all nodes in "/okm:trash/{userId}".
 For example, if the method is executed by the "okmAdmin" user, then the purged trash will be "/okm:trash/okmAdmin".		
 When a node is purged, it can only be restored from a previous repository backup. The purge action permanently removes the node from the repository.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            ws.repository.purgeTrash();
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getUpdateMessage

Description:

Method	Return values	Description
getUpdateMessage()	String	Retrieves a message when a new OpenKM release is available.

There is an official OpenKM update message service available that is based on your local OpenKM version.


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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getUpdateMessage());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

getRepositoryUuid

Description:

Method	Return values	Description
getRepositoryUuid()	String	Retrieves the installation's unique identifier.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                System.Console.WriteLine(ws.repository.getRepositoryUuid());  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

hasNode

Description:

Method	Return values	Description
hasNode(String nodeId)	Boolean	Returns a boolean indicating whether a node exists.

The value of the parameter nodeId can be a valid **UUID** or **path**.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Exists node:" + ws.repository.hasNode("064ff51a-b815-4f48-a096-b4946876"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getNodePath

Description:

Method	Return values	Description
getNodePath(String uuid)	String	Converts a node UUID to its path.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getNodePath("e339f14b-4d3a-489c-91d3-05e4575709d2"));
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(String path)	String	Converts a node path to a UUID.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getNodeUuid("/okm:root/tmp"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getAppVersion

Description:

Method	Return values	Description
getAppVersion()	AppVersion	Returns information about the OpenKM version.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getAppVersion().toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

copyAttributes

Description:

Method	Return values	Description
copyAttributes(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes)	void	Copies attributes from one node to another.

The value of the dstId parameter should be a node UUID or a path.



- When the categories parameter is true, the original values of the categories will be copied.
- When the keywords parameter is true, the original values of the keywords will be copied.
- When the propertyGroups parameter is true, the original values of the metadata groups will be

copied.

- When the notes parameter is true, the original values of the notes will be copied.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.repository.copyAttributes("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executeScript

Description:

Method	Return values	Description
executeScript(FileStream fs)	ScriptExecutionResult	Executes a script.
The local script - test.bsh - used in the sample below:		
<pre>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"); } }</pre>		

```

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

    // Some value can also be returned as string
    return "test result";

```



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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream("E:\\test.bsh", FileMode.Open);
                ScriptExecutionResult result = ws.repository.executeScript(fs);
                System.Console.WriteLine(result.result);
                System.Console.WriteLine(result.stdout);

                if (!result.stderr.Equals(""))
                {
                    System.Console.WriteLine("Error happened");
                    System.Console.WriteLine(result.stderr);
                }

                fs.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

executeScript

Description:

Method	Return values	Description
executeScript(String script)	ScriptExecutionResult	Executes a script.

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(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);
                Console.WriteLine(result.result);
                Console.WriteLine(result.stdout);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

executeSqlQuery

Description:

Method	Return values	Description
executeSqlQuery(FileStream fs)	SqlQueryResults	Executes SQL statements.

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 contain a single SQL statement.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream("E:\\test.sql", FileMode.Open);
                SqlQueryResults result = ws.repository.executeSqlQuery(fs);
                fs.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executeSqlQuery

Description:

Method	Return values	Description
executeSqlQuery(String sql)	SqlQueryResults	Executes SQL statements.
 The SQL script can only contain a single SQL statement. This action can only be done by a superuser (user with ROLE_ADMIN).		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                SqlQueryResults result = ws.repository.executeSqlQuery("SELECT NBS_UUID, NBS_CONTEXT, NBS_NAME");
                foreach (SqlQueryResultColumns columns in result.sqlQueryResults)
                {
                    Console.WriteLine("UUID:" + columns.sqlQueryResultColumns[0]);
                    Console.WriteLine("Context:" + columns.sqlQueryResultColumns[1]);
                    Console.WriteLine("Name:" + columns.sqlQueryResultColumns[2]);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executeHqlQuery

Description:

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

executeHqlQuery(FileStream fs)	HqlQueryResults	Executes HQL statements.
---------------------------------------	------------------------	--------------------------

The test.sql script used in the sample below:

```
SELECT uuid, name from NodeBase where name = 'okm:root';
```



The HQL script can only contain a single HQL statement.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream(@"C:\Desktop\test.sql", FileMode.Open);
                HqlQueryResults result = ws.repository.executeHqlQuery(fs);
                fs.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executeHqlQuery

Description:

Method	Return values	Description

executeHqlQuery(String hql)	HqlQueryResults	Executes HQL statements.
------------------------------------	------------------------	--------------------------



The HQL script can only contain a single HQL statement.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                HqlQueryResults result = ws.repository.executeHqlQuery("SELECT uuid, name from NodeBase where name like '%"+args[0]+"%'");
                foreach (HqlQueryResultColumns row in result.hqlQueryResults)
                {
                    Console.WriteLine("uuid: " + row.hqlQueryResultColumns[0] + ", name: " + row.hqlQueryResultColumns[1]);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getTranslations

Description:

Method	Return values	
getTranslations(String lang, String module)	Dictionary<String, String>	Retrieve



The OpenKM translations tables can be used to retrieve the current OpenKM translations or to create your own translations.

By default, module values are:

- frontend (used by the default OpenKM UI).
- extension (used by the OpenKM extension UI).
- mobile (used by the OpenKM mobile UI).

Example to add a new translation module:

SQL statements 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 should then be:

```
Dictionary<String, String> translations = ws.repository.getTranslations("en-GB", "doc");
```

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<string,string>translations = ws.repository.getTranslations("en-GB", "frontend");
                foreach (KeyValuePair<string, string> kvp in translations)
                {
                    Console.WriteLine("key:{0},with translation:{1}", kvp.Key, kvp.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getConfiguration

Description:

Method	Return values	Description
getConfiguration(String key)	Configuration	Retrieves the value of a configuration parameter.



If your OpenKM version has the configuration parameter named "**webservices.visible.properties**", access to configuration parameters will be restricted for non-administrator users. That means any non-administrator user who tries to access, via the webservices, configuration parameters not included in the list of values of "**webservices.visible.properties**" will receive an access denied exception.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Configuration configuration = ws.repository.getConfiguration("system.ocr");
                System.Console.WriteLine(configuration.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getChangeLog

Description:

Method	Return values	Description

getChangeLog(String nodePath, DateTime modificationsFrom)	List<ChangeLogged>	Returns the list of changes in a path and its subfolders.
--	---------------------------------	---



- The method is used by the desktop synchronization application to retrieve changes.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<ChangeLogged> changeLoggedList = ws.repository.getChangeLog("/okm:root/test", DateTime.Now);
                foreach (var cl in changeLoggedList)
                {
                    System.Console.WriteLine(cl.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getServerTime()

Description:

Method	Return values	Description
getServerTime()	String	Returns the current server time.



The server time returned is in ISO 8601 format.



The method is used by the desktop synchronization application to retrieve changes.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getServerTime());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getAvailableLocales

Description:

Method	Return values	Description
getAvailableLocales(String locale)	Dictionary<String, String>	Returns the available languages.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> locales = ws.repository.getAvailableLocales("en-GB");
                foreach(KeyValuePair<String, String> local in locales)
                {
                    Console.WriteLine("Language:" + local.Key + "," + local.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getLicenseInfo

Description:

Method	Return values	Description
getLicenseInfo()	LicenseInfo	Returns license information.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try

```

```

        {
            ws.login(user, password);
            Console.WriteLine(ws.repository.getLicenseInfo().ToString());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getClusterUuid

Description:

Method	Return values	Description
getClusterUuid()	String	Returns the cluster UUID.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

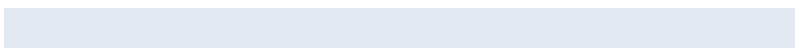
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Console.WriteLine("Cluster UUID =" + ws.repository.getClusterUuid());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getIsFilePlan

Description:



Method	Return values	Description
getIsFilePlan()	bool	True when the filePlan mode is active.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Console.WriteLine("FilePlan= " + ws.repository.getIsFilePlan().ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createShortenUrl

Description:

Method	Return values	Description
createShortenUrl(String name, String url)	String	Creates a shortened URL. The name parameter is optional (auto-generated if null). Returns the shortened URL.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String shortenUrl = ws.repository.createShortenUrl(null, "https://www.example.com/very/long/url");
                Console.WriteLine("shortenUrl");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Report samples

Basics

The table below shows how variables should be passed based on field type:

Field type	Type	Description
Date	String	Use the pattern yyyy-MM-dd (year - month - day) <code>2018-10-04</code>
Select multiple	String	Use "," to split each value. <code>"value1,value2,value3"</code>

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in to the webservice, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Report methods from the "**report**" class as shown below:

```
ws.report.getReports(true)
```

Methods

getReports

Description:

Method	Return values	Description
getReports(boolean active)	List<Report>	Returns a list of reports.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Report> reports = ws.report.getReports(true);
                foreach (Report rep in reports)
                {
                    System.Console.WriteLine(rep.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getReport

Description:

Method	Return values	Description
getReport(long rpId)	Report	Returns a report.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Report> reports = ws.report.getReports(true);
                Report report = new Report();
                foreach (var rep in reports)
                {
                    if (rep.fileName.Equals("DocumentCheckout.rep"))
                    {
                        // Get report
                        report = ws.report.getReport(rep.id);
                        break;
                    }
                }
                System.Console.WriteLine(report.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

execute

Description:

Method	Return values	Description
execute(long rpId, Dictionary<String, String> params, String format, String uuid)	Stream	Returns a document resulting from executing a report.
 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; it is usually useful for reports executed from the user interface where the results depend on the selected node).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Report> reports = ws.report.getReports(true);
                Report report = new Report();
                foreach (var rep in reports)
                {
                    if (rep.fileName.Equals("DocumentCheckout.rep"))
                    {
                        // Get report
                        report = ws.report.getReport(rep.id);
                        break;
                    }
                }

                Dictionary<String, String> param = new Dictionary<String, String>();
                param.Add("from_date", "2016-01-01");
                param.Add("to_date", "2016-12-23");
                String uuid = "";
                Stream stream = ws.report.execute(report.id, Report.FORMAT_PDF, param, uuid);

                BeanHelper beanHelper = new BeanHelper();
                Byte[] data = beanHelper.ReadToEnd(stream);
                FileStream fileStream = new FileStream(@"C:\Desktop\out.pdf", FileMode.OpenOrCreate, FileAccess.F
                foreach (byte b in data)
                {
                    fileStream.WriteByte(b);
                }
                fileStream.Close();
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

executeSql

Description:

Method	Return values	Description
executeSql(long rpId)	Stream	Returns a document resulting from executing a report SQL.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long rpId = 1;
                Stream stream = ws.report.executeSql(rpId);
                FileStream file = new FileStream("C:\\activity.csv", FileMode.OpenOrCreate, FileAccess.ReadWrite);
                stream.CopyTo(file);
                file.Close();
                stream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getSqlReports

Description:

Method	Return values	Description
getSqlReports(boolean active)	List<Report>	Returns a list of SQL reports.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<SqlReport> sqlReports = ws.report.getSqlReports(true);
                foreach(SqlReport sqlr in sqlReports)
                {
                    System.Console.WriteLine(sqlr.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

save

Description:

Method

save(long rpId, Dictionary<String, String> _params, String format, String dstId, String docName, String uuid)



Available formats:

- Report.FORMAT_CSV
- Report.FORMAT_DOCX
- Report.FORMAT_HTML
- Report.FORMAT_ODT
- Report.FORMAT_PDF
- Report.FORMAT_RTF
- Report.FORMAT_TEXT

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

The parameter **docName** is used to specify the name of the saved document.

The parameter **uuid** is the UUID of a node (this parameter is optional; it is usually useful for reports executed from

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long repld = 1;
                Dictionary<String, String> param = new Dictionary<String, String>();
                param.Add("from_date", "2018-01-01");
                param.Add("to_date", "2019-12-30");
                String uuid = "";
                Document doc = ws.report.save(repld, param, Report.FORMAT_PDF, "8599eab7-ae61-4628-8010-1103");
                System.Console.WriteLine(doc.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

saveSql

Description:

Method	Return values	Description
saveSql(long rpId, Dictionary<String, String> _params, String dstId, String docName)	Document	Executes the report SQL and saves the resulting CSV document.

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



The parameter **docName** is the file name of the report.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long rpId = 1;
                String dstId = "7f64d889-1600-4cbc-8134-ef6fb62287ec";
                String docName = "activity.csv";
                Document document = ws.report.saveSql(rpId, new Dictionary<string, string>(), dstId, docName);
                System.Console.WriteLine(document.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

generateDownloadReportToken

Description:

Method	Return values	Description
generateDownloadReportToken(long rpId)	String	Return the token for downloading the report.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long repld = 1;
                String dwtd = ws.report.generateDownloadReportToken(repld);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Stamp samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Stamp methods from the "**stamp**" class" as shown below:

```
ws.stamp.getAllStamps();
```

Methods

getAllStamps

Description:

Method	Return values	Description
getAllStamps()	List<StampItem>	Returns a list of stamp items.

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    List<StampItem> stamps = ws.stamp.getAllStamps();
    foreach(StampItem stamItem in stamps)
    {
        System.Console.WriteLine(stamItem.name);
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

testGetStampTextByPk

Description:

Method	Return values	Description
getStampTextByPk(long id, String uuid)	StampText	Returns the stamp of type Text by ID.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = 1;
                List<StampItem> stamps = ws.stamp.getStampTextByPk(id, document.uuid);
                System.Console.WriteLine(stamp.name);
                System.Console.WriteLine(stamp.description);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

    }
  }
}

```

getStampImageByPk

Description:

Method	Return values	Description
getStampImageByPk(long id, String uuid)	StampImage	Returns the stamp of type Image by ID.

Example:

```

using System;
using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = 1;
                List<StampItem> stamps = ws.stamp.getStampImageByPk(id, document.uuid);
                System.Console.WriteLine(stampImage.name);
                System.Console.WriteLine(stampImage.description);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getStampBarcodeByPk

Description:

Method	Return values	Description
getStampBarcodeByPk(long id, String uuid)	StampBarcode	Returns the stamp of type Barcode by ID.

Example:

```
using System;
using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = 1;
                StampBarcode stampBarcode = ws.stamp.getStampBarcodeByPk(id, "6330d2a0-529f-4c14-baa1-ce6a92");
                System.Console.WriteLine(stampBarcode.name);
                System.Console.WriteLine(stampBarcode.description);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

calculateStampCoordinates

Description:

Method	Return values	Description
calculateStampCoordinates(long imgToStampWidth, long imgToStampHeight, long floatingDivWidth, long floatingDivHeight, String exprX, String exprY, String stampType, String stampAlign)	StampCoordinates	Returns the calculated stamp coordinates.
 In the Expr. X and Expr. Y input fields, you can enter more than a simple number. Currently, the following macros are defined: • IMAGE_WIDTH • IMAGE_HEIGHT		

- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

To center a stamp on the page, you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

The parameter **stampAlign**.



Available values:

- text
- image

The parameter **stampAlign** is the text alignment.



Available values:

- left
- center
- right

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Parameters
            }
        }
    }
}
```

```

        long imgToStampWidth = 100;
        long imgToStampHeight = 100;
        long floatingDivWidth = 250;
        long floatingDivHeight = 300;
        string exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
        string exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
        string stampType = "text";
        string stampAlign = "center";
        StampCoordinates stampCoordinates = ws.stamp.calculateStampCoordinates(imgToStampWidth, imgToStampHeight,
        floatingDivWidth, floatingDivHeight, exprX, exprY, stampType, stampAlign);
        System.Console.WriteLine("X = " + stampCoordinates.x + ", Y = " + stampCoordinates.y);
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

calculateStampExpressions

Description:

Method	Return values	Description
calculateStampExpressions(long imgToStampWidth, long imgToStampHeight, long posX, long posY)	StampExpressions	Returns the calculated stamp expressions for X and Y.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Parameters
                long imgToStampWidth = 100;
                long imgToStampHeight = 100;
                long posX = 10;
                long posY = 50;
                StampExpressions stampExpressions = ws.stamp.calculateStampExpressions(imgToStampWidth, imgToStampHeight,
                posX, posY, "PAGE_CENTER - IMAGE_WIDTH / 2", "PAGE_MIDDLE - IMAGE_HEIGHT / 2", "text", "center");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

        System.Console.WriteLine("X = " + stampExpressions.exprX + ", Y = " + stampExpressions.exprY);
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

stampText

Description:

Method	Return values	Description
stampText(String uuid, long id)	void	Stamps text in a document.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = "1";
                ws.stamp.stampText("babe24df-c443-49a5-9453-39dfc9b27924", id);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

stampImage

Description:

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

stampImage(String uuid, long id)	void	Stamps an image in a document.
---	-------------	--------------------------------

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long id = "3";
                ws.stamp.stampImage("babe24df-c443-49a5-9453-39dfc9b27924", id);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

stampBarcode

Description:

Method	Return values	Description
stampBarcode(String uuid, long id)	void	Stamps a barcode in a document.

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
```

```

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

try
{
    ws.login(user, password);
    long id = 1;
    ws.stamp.stampBarcode("6330d2a0-529f-4c14-baa1-ce6a92004e01", id);
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

stampImageManually

Description:

Method	Return values	Description
stampImageCustom(String uuid, long id, String exprX, String exprY, String range, String personalStampUuid)	void	Stamps a document with a custom image.
The id is the ID of an image stamp which is created by the administrator. The exprX is the expression to set the position in X coordinates. The exprY is the expression to set the position in Y coordinates. The personalStampUuid must be a valid document UUID. The document must be an image (jpg or png).		
i In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined: • IMAGE_WIDTH • IMAGE_HEIGHT • PAGE_WIDTH • PAGE_HEIGHT • PAGE_CENTER • PAGE_MIDDLE So to center a stamp in the page you can use: Expr. X PAGE_CENTER		

	Expr. Y	PAGE_MIDDLE	
--	---------	-------------	--

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long stampId = "1";
                string exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
                string exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
                ws.stamp.stampImageCustom("babe24df-c443-49a5-9453-39dfc9b27924", stampId, exprX, exprY, "", "92
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

stampTextCustom

Description:

Method	Return values	Description
stampTextCustom(String uuid, long id, String text, String exprX, String exprY, String range)	void	Stamp in a document with a custom text.



The **id** is the ID of a text stamp which is created by the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.



In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long stampId = "1";
                string text = "Any text";
                string exprX = "PAGE_CENTER";
                string exprY = "PAGE_MIDDLE";
                ws.stamp.stampTextCustom("babe24df-c443-49a5-9453-39dfc9b27924", stampId, text, exprX, exprY, "");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

stampImageCustom

Description:

Method	Return values	Description
stampImageCustom(String uuid, long id, String exprX, String exprY, String range, double ratio, FileStream fs)	void	Stamp in a document with a custom image.

i The **id** is the ID of an image stamp which is created by the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

i In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

    try
    {
        ws.login(user, password);
        long stampId = "1";
        string exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
        string exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
        FileStream fsimg = new FileStream(@"C:\test.jpg", FileMode.Open, FileAccess.Read);
        ws.stamp.stampImageCustom("babe24df-c443-49a5-9453-39dfc9b27924", stampId, exprX, exprY, "", 0, fsi
        fsimg.Close();
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

calculateFlyImageDimensions

Description:

Method	Return values	Description
calculateFlyImageDimensions(String uuid, long imgToStampWidth, long imgToStampHeight, long floatingImageWidth, long floatingImageHeight, int pageNumber)	StampImageSize	Return the calculated image to stamp height and width size.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Parameters
                long imgToStampWidth = 100;
                long imgToStampHeight = 100;
                long floatingImageWidth = 300;
                long floatingImageHeight = 300;
                int pageNumber = 2;
            }
        }
    }
}

```

```

        StampImageSize stampImageSize = ws.stamp.calculateFlyImageDimensions("babe24df-c443-49a5-9453
        floatingImageWidth, floatingImageHeight, pageNumber);
        System.Console.WriteLine(stampImageSize);
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}
}

```

getPersonalStamps

Description:

Method	Return values	Description
getPersonalStamps()	List<Document>	Returns a list of personal seals.

 Returns a list of documents of type image (jpg or png) from /okm:templates/userId/stamps.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Document> docs = ws.stamp.getPersonalStamps();
                foreach (Document doc in docs)
                {
                    System.Console.WriteLine(doc.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```

    }
  }
}

```

getPersonalStampImage

Description:

Method	Return values	Description
getPersonalStampImage(String uuid, long id)	StampPersonalImage	Returns the processed personal seal.



When stamping a document with an image, the image can also have a text layer.

This method processes the image and returns an image with the text layer.

The **uuid** must be a **UUID** returned by the method `getPersonalStamps`.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long stampId = 1;
                StampPersonalImage stampPersonalImage = ws.stamp.getPersonalStampImage("928de61e-6ceb-4f94-b
                System.Console.WriteLine(stampPersonalImage.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

Search samples

Basics

Most methods use QueryParams. Here are some tips about how to use them.

Variables	Type	Allow wildcards	Remarks
domain	long	No.	Available values: • QueryParams.DOCUMENT • QueryParams.FOLDER • QueryParams.MAIL • QueryParams.RECORD By default, the value is set to QueryParams.DOCUMENT. For searching documents and folders, use the value: (QueryParams.DOCUMENT QueryParams.FOLDER)
author	String	No.	The value must be a valid userId.
name	String	Yes.	
title	String	Yes.	
keywords	List<String>	Yes.	
categories	List<String>	No.	Values should be category UUIDs; do not use path values.
content		Yes.	

mimeType		No.	The value should be a valid and registered MIME type. Only applies to document nodes.
language		No.	The value should be a valid language. Only applies to document nodes.
folder		No.	When empty, the "/okm:root" node is used by default. The value should be a valid UUID; do not use a path value.
folderRecursive	Boolean	No.	It only makes sense to set this variable to true when the folder var
lastModifiedFrom	Calendar	No.	
lastModifiedTo	Calendar	No.	
mailSubject	String	Yes.	Only applies to mail nodes.

mailFrom	String	Yes.	Only applies to mail nodes.
mailTo		Yes.	Only applies to mail nodes.
notes		Yes.	
properties	Dictionary<String, String>	Yes, on almost all.	Wildcards cannot be applied to metadata field values like "date". The dictionary of the properties is composed of pairs: ('metadata_field_name','metadata_field_value') For example: <pre>Dictionary<String, String> properties = new Dictionary<String, String>(); properties.Add("okp:consulting.name", "name value")</pre> Filtering by a range of dates: <pre>DateTime date = DateTime.Now; // today DateTime to = new DateTime(date.Year, date.Month, date.Day, 23, 59, 59); DateTime from = to.AddDays(-3); // three days before Dictionary<String, String> properties = new Dictionary<String, String>(); properties.Add("okp:consulting.date", ISO8601.formatBa</pre>  When filtering by a range of dates, you must specify the date range in the query, otherwise the filter will be ignored by OpenKM. To filter by a metadata field of type multiple like <pre><select label="Multiple" name="okp:consulting.multiple" type="text"> <option label="One" value="one"/> <option label="Two" value="two"/> <option label="Three" value="three" /> </select></pre> You should do:

```
properties.Add("okp:consulting.multiple", "one;two");
```

Where **"one"** and **"two"** are valid values and the ch



The search operation is performed using only AND logic.

Wildcard examples:

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

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the search methods from the **"search"** class as 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 - the marshalling and unmarshalling process - while you are only interested in using a few object variables. If that's your case, you can use NodeProperties classes to retrieve only the object variables that you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams qParams = new QueryParams();
                qParams.domain = QueryParams.DOCUMENT;
                qParams.name = "test*.html";
                foreach (QueryResult qr in ws.search.find(qParams, null))
                {
                    System.Console.WriteLine(qr);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

find

Description:

Method	Return values	Description
find(QueryParams queryParams, String sortField, bool 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 - the marshalling and unmarshalling process - while you are only interested in using a few object variables. If it's your case, you can use NodeProperties classes to retrieve only the object variables that you really need.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams _params = new QueryParams();
                _params.domain = QueryParams.DOCUMENT + QueryParams.FOLDER;
                _params.name = "test*";
                foreach (QueryResult qr in ws.search.find(_params, SearchSortField.LAST_MODIFIED, true, null))
            }
        }
    }
}

```

```

        {
            Console.WriteLine(qr.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

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 approximate values because they depend on hardware and the specific scenario where the application is running). The other search methods return objects of type Node, which include full node data; the SimpleNodeBase object provides less data but in most cases should be enough. Especially when the limit is a higher value, you will appreciate the difference in performance.



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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" skips that many results before returning results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams qParams = new QueryParams();
                qParams.domain = QueryParams.DOCUMENT;
                qParams.name = "test*";
                SimpleNodeBaseResultSet snbrs = ws.search.findSimpleNodeBasePaginated(qp, 0, 10);
                System.Console.WriteLine("Total results:" + snbrs.total);

                foreach (SimpleNodeBase snb in snbrs.results)
                {
                    System.Console.WriteLine(snb.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

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 approximate values because they depend on hardware and the specific scenario where the application is running). The other search methods return objects of type `Node`, which include full node data; the `SimpleNodeBase` object provides less data but in most cases should be enough. Especially when the limit is a higher value, you will appreciate the difference in performance.



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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" skips that many results before returning 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:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
using com.openkm.sdk4csharp.impl;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
        }  
    }  
}
```

```

String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    QueryParams qParams = new QueryParams();
    qParams.domain = QueryParams.DOCUMENT;
    qParams.name = "test*";
    SimpleNodeBaseResultSet snbrs = ws.search.findSimpleNodeBasePaginated(params, SearchSortField.A
    System.Console.WriteLine("Total results:" + snbrs.total);

    foreach (SimpleNodeBase snb in snbrs.results)
    {
        System.Console.WriteLine(snb.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

findSimpleNodeBasePaginated

Description:

Method	Return values	Description
findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, bool sortReverse, int offset, int limit, String pluginName)	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 approximate values because they depend on hardware and the specific scenario where the application is running). The other search methods return objects of type Node, which include full node data; the SimpleNodeBase object provides less data but in most cases should be enough. Especially when the limit is a higher value, you will appreciate the difference in 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams qp = new QueryParams();
                qp.name = FOLDER_NAME1 + "***";
                qp.domain = QueryParams.FOLDER;
                SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBasePaginated (qp, SearchSortField.AUTHOR);
                Console.WriteLine("Total: " + result.total);
                foreach (SimpleNodeBase nodeBase in result.results)
                {
                    if (nodeBase != null)
                    {
                        Console.WriteLine(nodeBase.toString());
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

```

    }
  }
}

```

getCategorizedDocuments

Description:

Method	Return values	Description
getCategorizedDocuments(String categoryId)	List<Document>	Retrieves a list of all documents related to a category.
The values of the categoryId parameter should be a category folder UUID or a path.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Document doc in ws.search.getCategorizedDocuments("50b7a5b9-89d2-430e-bbc9-6a6e01662a"))
                {
                    System.Console.WriteLine(doc);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

saveSearch

Description:

Method	Return values	Description
saveSearch(QueryParams params)	Long	Saves search parameters.
The queryName variable of the params parameter must be initialized.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams qParams = new QueryParams();
                qParams.domain = QueryParams.DOCUMENT;
                qParams.name = "test*.html";

                foreach (QueryResult qr in ws.search.find(qParams))
                {
                    System.Console.WriteLine(qr);
                }

                // Save the search to be used later
                qParams.queryName = "sample search";
                ws.search.saveSearch(qParams);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

updateSearch

Description:

Method	Return values	Description
updateSearch(QueryParams params)	void	Updates previously saved search parameters.



It can only be updated if it was created by the same user who is executing the method.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryParams qParams in ws.search.getAllSearchs())
                {
                    if (qParams.queryName.Equals("sample search"))
                    {
                        // Change some value.
                        qParams.name = "admin*.html";
                        ws.search.updateSearch(qParams);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getSearch

Description:

Method	Return values	Description
getSearch(int qpId)	QueryParams	Gets saved search parameters.



It can only be retrieved if it was created by the same user who is executing the method.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int qpld = 1; // Some valid search id
                QueryParams qParams = ws.search.getSearch(qpld);
                System.Console.WriteLine(qParams);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getAllSearches

Description:

Method	Return values	Description
getAllSearches()	List<QueryParams>	Retrieves a list of all saved search parameters.
 It can only retrieve the list of saved searches created by the same user who is executing the method.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryParams qParams in ws.search.getAllSearches())
                {
                    System.Console.WriteLine(qParams.queryName);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

deleteSearch

Description:

Method	Return values	Description
deleteSearch(int qpId)	void	Deletes saved search parameters.
 It can only be deleted if it was created by the same user who is executing the method.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            int qpId = 1; // Some valid search id
            ws.search.deleteSearch(qpId);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getSearchConfig

Description:

Method	Return values	Description
getSearchConfig(String pluginName)	NodeSearchConfig	Returns a NodeSearchConfig object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                NodeSearchConfig nodeSearchConfig = ws.search.getSearchConfig("com.openkm.plugin.search.TestNo
                Console.WriteLine(nodeSearchConfig.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getSearchPlugins

Description:

Method	Return values	Description
getSearchPlugins()	SearchPluginList	Returns a SearchPluginList object.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                SearchPluginList pluginList = ws.search.getSearchPlugins();
                foreach (SearchPlugin searchPlugin in pluginList.searchPlugins)
                {
                    Console.WriteLine(searchPlugin.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

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.

The **syntax** to use in the statement parameter is the pair '**field:value**'. For example:



- "name:grial" filters the name field by the word "grial".

More information about Lucene syntax 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.



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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                SimpleNodeBaseResultSet snbr = ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND nar

                foreach (SimpleNodeBase nodeBase in snbr.results)
                {
                    if (nodeBase != null)
                    {
                        System.Console.WriteLine(nodeBase.toString());
                    }
                }
            }
        }
    }
}
```

```

    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

findSimpleNodeBaseByQueryPaginated

Description:

Method	Return values	Description
findSimpleNodeBaseByQueryPaginated(String query, String sortField, bool sortReverse, int offset, int limit)	SimpleNodeBaseResultSet	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.



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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                SimpleNodeBaseResultSet snbr= ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND nan

                foreach (SimpleNodeBase nodeBase in snbr.results)
                {
                    if(nodeBase != null)
                    {
                        System.Console.WriteLine(nodeBase.toString());
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findByQuery

Description:

Method	Return values	Description
findByQuery(String query, 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 - marshaling and unmarshaling process - while you are only interesting on using only a few Objects variables. If it's your case you can use `NodeProperties` classes for retrieving the Object variables what you really need.

The **syntax** to use in the statement parameter is the pair '**field:value**'. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at [Lucene query syntax](#).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryResult qr in ws.search.findByQuery("keyword:test AND name:t*.pdf", null))
                {
                    System.Console.WriteLine(qr);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findByQuery

Description:

Method	Return values	Description
findByQuery(String query, String sortField, bool		Returns a list of results

sortReverse, String propertiesPlugin)	List<QueryResult>	filtered by the query parameter.
 The syntax to use in the statement parameter is the pair 'field:value'. For example: "name:grial" filters the name field by the word "grial". More information about Lucene syntax is available at Lucene query syntax.		
 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryResult qr in ws.search.findByQuery("keyword: test", SearchSortField.NAME, true, null))
                {
                    System.Console.WriteLine(qr);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findAllDefaultByNodeClass

Description:

Method	Return values	Description
findAllDefaultByNodeClass(long ncId)	List<QueryParams>	Retrieves a list of saved searches for a NodeClass.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long ncId = 1;
                List<QueryParams> queryParams = ws.search.findAllDefaultByNodeClass(ncId);
                foreach (QueryParams qParams in queryParams)
                {
                    System.Console.WriteLine(qParams.queryName);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findWithMetadata

Description:

Method	Return values	Description
findWithMetadata(QueryParams _params, 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 the 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 — the marshalling and unmarshalling process — while you are only interested in using a few object variables. If that's your case, you can use NodeProperties classes to retrieve the object variables that you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // QueryParams
                QueryParams qp = new QueryParams();
                qp.folderRecursive = true;
                qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryParams.MAIL;
                qp.name = "test*";

                // Properties
                Dictionary<string, string> properties = new Dictionary<string, string>();
                properties.Add("okp:consulting.name", "test");
                qp.properties = properties;

                // Groups
                List<string> groups = new List<string>();
                groups.Add("okg:consulting");

                List<QueryResult> queryResults = ws.search.findWithMetadata(qp, null, groups);
                foreach (QueryResult qResult in queryResults)
                {
                    System.Console.WriteLine(qResult.toString());
                }
            }
            catch (Exception e)
            {
            }
        }
    }
}
```

```

    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

findWithMetadata

Description:

Method	Return values	Description
findWithMetadata(QueryParams queryParams, String sortField, bool 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 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

```



Retrieving entire objects (Document, Folder, Record, Mail) from REST can take a lot of time — the marshalling and unmarshalling process — while you are only interested in using a few object variables. If that's your case, you can use NodeProperties classes to retrieve the object variables that you really need.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {

```

```

static void Main(string[] args)
{
    String host = "http://localhost:8080/openkm";
    String username = "okmAdmin";
    String password = "admin";
    OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

    try
    {
        ws.login(user, password);
        // QueryParams
        QueryParams qp = new QueryParams();
        qp.folderRecursive = true;
        qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryPara
        qp.name = "test*";

        // Properties
        Dictionary<string, string> properties = new Dictionary<string, string>();
        properties.Add("okp:consulting.name", "test");
        qp.properties = properties;

        // Groups
        List<string> groups = new List<string>();
        groups.Add("okg:consulting");

        List<QueryResult> queryResults = ws.search.findWithMetadata(qp, SearchSortField.NAME, true, null, grc
        foreach (QueryResult qr in queryResults)
        {
            Console.WriteLine(qr.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

findWithMetadataPaginated

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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before starting to return results.
- 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 to 20, you should use these values:

- `limit=10`
- `offset=10`

Retrieving entire objects (Document, Folder, Record, Mail) from REST can take a lot of time — the marshalling and unmarshalling process — while you are only interested in using a few object variables. If that's your case, you can use `NodeProperties` classes to retrieve the object variables that you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // QueryParams
                QueryParams qp = new QueryParams();
                qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryParams.MAIL;
                qp.name = "test*";

                // Properties
                Dictionary<string, string> properties = new Dictionary<string, string>();
                properties.Add("okp:consulting.name", "test");
                qp.properties = properties;

                // Groups
                List<string> groups = new List<string>();
                groups.Add("okg:consulting");
            }
        }
    }
}
```

```

        ResultSet resultSet = ws.search.findWithMetadataPaginated(qp, 0, 10, null, groups);
        System.Console.WriteLine("Total results: " + rs.total.ToString());
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

findWithMetadataPaginated

Description:

Method	Return values	Description
findWithMetadataPaginated(QueryParams queryParams, String sortField, bool sortReverse, 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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

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

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.

i Available **sortField** values:

```

SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED

```

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 to 20, you should use these values:

- limit=10
- offset=10

Retrieving entire objects (Document, Folder, Record, Mail) from REST can take a lot of time — the marshalling and unmarshalling process — while you are only interested in using a few object variables. If that's your case, you can use `NodeProperties` classes to retrieve the object variables that you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // QueryParams
                QueryParams qp = new QueryParams();
                qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryParams.MAIL;
                qp.name = "test*";

                // Properties
                Dictionary<string, string> properties = new Dictionary<string, string>();
                properties.Add("okp:consulting.name", "test");
                qp.properties = properties;

                // Groups
                List<string> groups = new List<string>();
                groups.Add("okg:consulting");

                ResultSet rs = ws.search.findWithMetadataPaginated(qp, SearchSortField.AUTHOR, true, 0, 10, null, groups);
                Console.WriteLine("Total results: " + rs.total.ToString());
                foreach (QueryResult qr in rs.results)
                {
                    Console.WriteLine(qr.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getMimeTypes

Description:

Method	Return values	Description
getMimeTypes()	List<MimeType>	Retrieves a list of MIME types.

 Only the list of MIME types that may be used in the search will be retrieved.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (MimeType mimeType in ws.search.getMimeTypes())
                {
                    System.Console.WriteLine(mimeType.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

csvExport

Description:

Method	Return values	Description

**csvExport(String lang, QueryParams queryParams,
List<String> propertyGroups, bool compact)**

InputStream

Export as a csv a list of results filtered by the values of the queryParams parameter.

The parameter **lang** must be ISO 639-1 compliant.



More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams _params = new QueryParams();
                _params.domain = QueryParams.DOCUMENT + QueryParams.FOLDER;
                _params.name = "test*";

                List <string > propertyGroups = new List<string>();
                propertyGroups.Add("okg:consulting");
                propertyGroups.Add("okg:technology");

                Stream stream = ws.search.csvExport("es-ES", _params, propertyGroups, false);
                FileStream destFile = new FileStream("C:\\Testing\\export.csv", FileMode.OpenOrCreate);
                stream.CopyTo(destFile);
                destFile.Dispose();
                stream.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Shard samples

Basics



Example of UUID:

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

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in to the web service, the session is kept in the web service object. Then you can use the other API methods.

At this point you can use all the Shard methods from the "**shard**" class as shown below:

```
ws.shard.getShards()
```

Methods

getShards

Description:

Method	Return values	Description
getShards()	List<Shard>	Returns a list of all shards.

Example:

```
using System; using System.Collections.Generic;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach(bean.Shard shard in ws.shard.getShards())
                {
                    Console.WriteLine(shard.ToString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setCurrentShard

Description:

Method	Return values	Description
setCurrentShard(long shardId)	void	Changes the current shard.

Example:

```

using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long shardId = 1;
                ws.shard.setCurrentShard(shardId);
            }
        }
    }
}

```

```
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}
```

changeShard

Description:

Method	Return values	Description
changeShard(String uuid, long shardId)	void	Changes the shardId of the UUID.

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long shardId = 1;
                ws.shard.changeShard("e2391b01-ce10-42c7-94a9-b0d6b394048e", shardId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Task samples

Basics

Task fields description:

Field	Type	Description	Mandatory
Id	long	Internal task ID.  It is automatically set by the application.	Not applicable.
Owner	String	It contains the OpenKM user ID.  It is automatically set by the application.	Not applicable.
Subject	String	The topic of the task.	Yes.
Description	String	The description of the task.	No.
Start	DateTime	When the task might start.	Yes.
End	DateTime	When the task might end.	No.
Status	TaskStatus	The task status.  You administer the list of status values.	Yes.
Project	TaskProject	The project related to the task.  You administer the list of project values.	Yes.

Type	TaskType	The task type.  You administer the list of task type values.	Yes.
Progress	int	The numeric progress status.  The range of allowed values is from 0 to 100, where 100% indicates a completed task.	Yes.
RepeatGroup	long	When a task is repeated over time it is a member of a group. The group is identified by a unique repeat group ID.	No.
ReminderStartValue	int	How many days before the task starts the system might send an email notification to the user.	No.
ReminderEndValue	int	How many days before the task ends the system might send an email notification to the user.	No.
ReminderStartUnit	String	Reminder start units.  Allowed values: • "m" for minutes. • "h" for hours. • "d" for days.	Mandatory when ReminderStartValue is greater than 0.
ReminderEndUnit	String	Reminder end units.  Allowed values: • "m" for minutes.	Mandatory when ReminderEndValue is greater than 0.

		• "h" for hours. • "d" for days.	
Users	List<String>	Collection of users assigned to the task.	No. But at least one user or role should be assigned.
Roles	List<String>	Collection of assigned roles to the task.	No. But at least one user or role should be assigned.
Documents	List<String>	List of UUIDs of the related documents.	No.
Folders	List<String>	List of UUIDs of the related folders.	No.
Mails	List<String>	List of UUIDs of the related mails.	No.
Records	List<String>	List of UUIDs of the related records.	No.

On all methods, you'll see a parameter named "**token**". When accessing the application via SOAP, the login process returns a token, which is used to identify the user on all exposed methods. From the default application execution context you must use the "**null**" value, which indicates the application must use the "**user session**".



In special cases, you might be "**promoted as Administrator**" using the "**administrator token**".

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

At this point you can use all the Task methods from the "task" class as shown below:

```
ws.task.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, subject);
```

Methods

getAssigned

Description:

Method	
getAssigned(long projectId, long typeId, long statusId, String orderColumn, bool orderAsc, int offset, int limit, DateTime fromDate, DateTime toDate, String subject)	i Filter parameters description: • The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id. • The statusId parameter must be a valid status id. • The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "id", "subject", "created", "modified", "assigned", "status", "type", "priority", "category", "tags", "location", "parent", "child", "subtask", "task", "project", "type", "status", "priority", "category", "tags", "location", "parent", "child", "subtask", "task". • The orderAsc parameter orders ascending or descending. • The from date to filter (this parameter is optional). • The to date to filter (this parameter is optional). • The subject parameter is the topic of the task. ✓ The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query. • The parameter "limit" is used to limit the number of results returned. • The parameter "offset" specifies how many results to skip before returning results. i For example, if your query has 1000 results, but you only want to return the first 10, you should use these values: • limit=10 • offset=0 Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long statusId = 1; // A valid status id
                String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
                DateTime? from = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1);
                DateTime? to = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 30);
                String subject = "test";

                TaskList assignedTasks = ws.task.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, subject);
                foreach (Task task in assignedTasks.tasks)
                {
                    System.Console.WriteLine(task.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getActive

Description:

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

getActive(long projectId, long typeId, long statusId, String orderColumn, bool orderAsc, int offset, int limit, DateTime? from, DateTime? to, String subject)	TaskList	Retrieve a list of active tasks assigned to a user, filtered and paginated.
		
Filter parameters description:		
• The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id. • The statusId parameter must be a valid status id. • The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation. • The orderAsc parameter orders ascending or descending. • The from date to filter (this parameter is optional). • The to date to filter (this parameter is optional). • The subject parameter is the topic of the task.		
		
The parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.		
• The parameter "limit" is used to limit the number of results returned. • The parameter "offset" specifies how many results to skip before returning results.		
		
For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:		
• limit=10 • offset=0		
Now suppose you want to show the results from 11-20, you should use these values:		
• limit=10 • offset=10		
		
The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebserviceFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long statusId = 1; // A valid status id
                String orderColumn = "subject"; // A valid TaskManagerTask class parameter name used in HQL query
                DateTime? from = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1);
                DateTime? to = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 30);
                String subject = "test";

                TaskList activeTasks = ws.task.getActive(projectId, typeId, statusId, "description", false, 0, 10, from, to, subject);
                foreach (Task task in activeTasks.tasks)
                {
                    System.Console.WriteLine(task.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getFinished

Description:

Method	Return values	Description
getFinished(long projectId, long typeId, long statusId, String orderColumn, bool orderAsc, int offset, int limit, DateTime? from, DateTime? to, String subject)	TaskList	Retrieve a list of finished tasks assigned to a user, filtered and paginated.

 Filter parameters description:

- The projectId parameter must be a valid project id.

- The typeId parameter must be a valid type id.
- The statusId parameter must be a valid status id.
- The orderColumn parameter must be a valid parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional).
- The to date to filter (this parameter is optional).
- The subject parameter is the topic of the task.



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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
        }
```

```

String host = "http://localhost:8180/openkm";
String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    long projectId = 1; // A valid project id
    long typeId = 1; // A valid type id
    long statusId = 1; // A valid status id
    String orderColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
    DateTime? from = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1);
    DateTime? to = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 30);
    String subject = "test";

    TaskList finishedTasks = ws.task.getFinished(projectId, typeId, statusId, "description", false, 0, 10, from, to);
    foreach (Task task in finishedTasks.tasks)
    {
        System.Console.WriteLine(task.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

getNotified

Description:

Method	Return values	Description
getNotified(long projectId, long typeId, long statusId, String orderColumn, bool orderAsc, int offset, int limit, DateTime? from, DateTime? to, String subject)	List<Task>	Retrieve a list of notified tasks assigned to a user, filtered and paginated.



Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameter must be a valid type id.
- The statusId parameter must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at [javadoc documentation](#).
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional).

- The to date to filter (this parameter is optional).
- The subject parameter is the topic of the task.



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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning results.



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

- limit=10
- offset=0

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

- limit=10
- offset=10



The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long statusId = 1; // A valid status id
                String orderByColumn = "start"; // A valid TaskManagerTask class parameter name used in HQL query
                DateTime? from = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 1);
```

```

        DateTime? to = new DateTime(DateTime.Now.Year, DateTime.Now.Month, 30);
        String subject = "test";

        TaskList taskList = ws.task.getNotified(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, subject);
        foreach (Task task in taskList.tasks)
        {
            Console.WriteLine(task.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getStatus

Description:

Method	Return values	Description
getStatus()	List<TaskStatus>	Retrieve a list of all task statuses.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<TaskStatus> list = ws.task.getStatus();
                foreach (TaskStatus taskStatus in list)
                {
                    System.Console.WriteLine(taskStatus.name);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}
```

getProjects

Description:

Method	Return values	Description
getProjects(bool filterActive)	List<TaskProject>	Retrieve a list of all the task projects.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<TaskProject> taskProjects = ws.task.geProjects(true);
                foreach (TaskProject tpj in taskProjects)
                {
                    System.Console.WriteLine(tpj.name);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getTypes

Description:

Method	Return values	Description
getTypes(bool filterActive)	List<TaskType>	Retrieve a list of all the task types.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<TaskType> taskTypes = ws.task.getTypes(false);
                foreach (TaskType tp in taskTypes)
                {
                    System.Console.WriteLine(tp.name);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getAssignedCount

Description:

Method	Return values	Description
getAssignedCount(long statusId, long projectId, long typeId)	long	Returns the number of tasks assigned to a user.
 Filter parameters description: • The statusId parameter must be a valid status id. • The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long statusId = 1; // A valid status id
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long assigned = ws.task.getAssignedCount(statusId, projectId, typeId);
                System.Console.WriteLine("Assigned tasks count: " + assigned.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getActiveCount

Description:

Method	Return values	Description
getActiveCount(long statusId, long projectId, long typeId)	long	Returns the number of active tasks for a user.
 Filter parameters description: • The statusId parameter must be a valid status id. • The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long statusId = 1; // A valid status id
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long active = ws.task.getActiveCount(statusId, projectId, typeId);
                System.Console.WriteLine("Active tasks count: " + active.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getFinishedCount

Description:

Method	Return values	Description
getFinishedCount(long statusId, long projectId, long typeId)	long	Returns the number of finished tasks for a user.
 Filter parameters description: • The statusId parameter must be a valid status id. • The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long statusId = 1; // A valid status id
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long finished = ws.task.getFinishedCount(statusId, projectId, typeId);
                System.Console.WriteLine("Finished tasks count: " + finished.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getNotifiedCount

Description:

Method	Return values	Description
getNotifiedCount(long statusId, long projectId, long typeId)	Long	Returns the number of notified tasks assigned to a user.
 Filter parameters description: • The statusId parameter must be a valid status id. • The projectId parameter must be a valid project id. • The typeId parameter must be a valid type id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id
                long statusId = 1; // A valid status id
                long total = ws.task.getNotifiedCount(statusId, projectId, typeId);
                Console.WriteLine("Total: " + total.ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

create

Description:

Method

create(String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> relatedDocuments, List<String> relatedFolders, List<String> relatedRecords, List<String> relatedMails, String repeatExpression, String repeatExpression, String formatDate, int repeatTimes, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)



The **repeatExpression** parameter description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of month, or day of week) matches the current time. The scheduler examines crontab entries

minute. The time and date fields are:

```

* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ???? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same)
? ? ? ?????????? month (1 - 12)
? ? ?????????????? day of month (1 - 31)
? ?????????????????? hour (0 - 23)
???????????????????? min (0 - 59)

```

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String subject = "task subject";

                DateTime date = DateTime.Now;
                String start = ISO8601.formatBasic(date);
                date = new DateTime(2019, 1, 1);
                String end = ISO8601.formatBasic(date);

                String description = "description test";
                long statusId = 1; // A valid task status id
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id

                List<String> notificationUsers = new List<string>();
                notificationUsers.Add("pherrera");
                notificationUsers.Add("sochoa");

                List<String> externalUsers = new List<String>();
                externalUsers.Add("test@openkm.com");

                List<String> relatedDocuments = new List<String>();
                relatedDocuments.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");

                List<String> relatedFolders = new List<String>();
                relatedFolders.Add("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");

                List<String> relatedMails = new List<String>();
            }
        }
    }
}

```

```
        List<String> relatedRecords = new List<String>();

        Task task = ws.task.create(subject, start, end, description, statusId, projectId, typeId, "okmAdmin", notificationUsers,
            relatedDocuments, relatedFolders, relatedRecords, relatedMails, "0 0 1 * *", "", "", 10, "m", 5, "m", 10);
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

update

Description:

Method	Return values
update(long taskId, String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> relatedDocuments, List<String> relatedFolders, List<String> relatedRecords, List<String> relatedMails, String owner, String repeatExpression, String repeatUntil, String formatDate, int repeatTimes, int progress, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)	Task

 The **repeatExpression** parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at least one of the two day fields (day of the month, or day of the week) matches the current time. The scheduler examines crontab once every minute. The time and date fields are:

```
***** command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ????? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)
? ? ? ????????? month (1 - 12)
? ? ????????????? day of month (1 - 31)
? ????????????????? hour (0 - 23)
???????????????????? min (0 - 59)
```

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String taskId = "2";
                String subject = "task update";

                DateTime date = DateTime.Now;
                String start = ISO8601.formatBasic(date);

                date = new DateTime(2019, 12, 1);
                String end = ISO8601.formatBasic(date);

                String description = "test update";
                long statusId = 1; // A valid task status id
                long projectId = 1; // A valid project id
                long typeId = 1; // A valid type id

                List<String> notificationUsers = new List<String>();
                roles.Add("pherrera");
                roles.Add("sochoa");

                List<String> externalUsers = new List<String>();
                externalUsers.Add("test@openkm.com");

                List<String> relatedDocuments = new List<String>();
                relatedDocuments.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");

                List<String> relatedFolders = new List<String>();
                relatedFolders.Add("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");

                List<String> relatedMails = new List<String>();
                List<String> relatedRecords = new List<String>();

                String owner = "okmAdmin";
                int progress = 20; // Progress task

                String repeatExpression = "0 0 1 * *";
                long taskId = 1;
                Task task = ws.task.update(taskId, subject, start, end, description, statusId, projectId, typeId, "okmAdmin",
                    relatedDocuments, relatedFolders, relatedRecords, relatedMails, owner, repeatExpression, "", "", progre
                System.Console.WriteLine(task.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}
```

getTask

Description:

Method	Return values	Description
getTask(long taskId)	Task	Retrieve a task.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8180/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long taskId = "2";  
                Task task = ws.task.getTask(task.id);  
                System.Console.WriteLine(task.toString());  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

delete

Description:

Method	Return values	Description
delete(long taskId)	void	Delete a task.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskId = "2";
                ws.task.delete(task.id);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

createProject

Description:

Method	Return values	Description
createProject(String name, bool active, String description)	TaskProject	Create a task project.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
```

```

String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    TaskProject taskProject = ws.task.createProject("Task project 001", true, "Any description");
    System.Console.WriteLine(taskProject.toString());
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}
}

```

updateProject

Description:

Method	Return values	Description
updateProject(long projectId, bool active, String name, String description)	TaskProject	Update a task project.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskProjectId = 1; // Valid task project id
                TaskProject taskProject = ws.task.updateProject(taskProjectId, true, "Task project 002", "Any description");
                System.Console.WriteLine(taskProject1.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

deleteProject

Description:

Method	Return values	Description
deleteProject(long projectId)	void	Delete a task project.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8180/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long taskProjectId = 1; // Valid task project id  
                ws.task.deleteProject(taskProjectId);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

getProject

Description:

Method	Return values	Description
getProject(long projectId)	TaskProject	Get a task project.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskProjectId = 1; // Valid task project id
                TaskProject tp = ws.task.getProject(taskProjectId);
                System.Console.WriteLine(tp.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

createType

Description:

Method	Return values	Description
createType(String name, bool active, String description)	TaskType	Create a task type.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";

```

```

        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            TaskType taskType = ws.task.createType("Test", false, "Any description");
            System.Console.WriteLine(taskType.toString());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

updateType

Description:

Method	Return values	Description
updateType(long typeId, bool active, String name, String description)	TaskType	Update a task type.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskTypeId = 1; // Valid task type id
                TaskType taskType = ws.task.updateType(taskTypeId, true, "New Test", "New description");
                System.Console.WriteLine(taskType.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

deleteType

Description:

Method	Return values	Description
deleteType(long typeId)	void	Delete a task type.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long typeId = 1; //Valid task type id
                ws.task.deleteType(taskTypeId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getType

Description:

Method	Return values	Description
getType(long typeId)	TaskType	Get a task type object by id.

Example:

```
using System;
using System.Collections.Generic;
```

```

using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskTypeId = 1; //Valid task type id
                TaskType taskType = ws.task.getType(taskTypeId);
                System.Console.WriteLine(taskType.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

createStatus

Description:

Method	Return values	Description
createStatus(String name, bool finish)	TaskStatus	Create a task status.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            TaskStatus taskStatus = ws.task.createStatus("Test", false);
            System.Console.WriteLine(taskStatus.toString());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

updateStatus

Description:

Method	Return values	Description
updateStatus(long statusId, String name, bool finish)	TaskStatus	Update a task status.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskStatusId = 1; // Valid task status id
                TaskStatus taskStatus = ws.task.updateStatus(taskStatusId, "Test", true);
                System.Console.WriteLine(taskStatus.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

deleteStatus

Description:

Method	Return values	Description
deleteStatus(long statusId)	void	Delete a task status.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskStatusId = 1; // Valid task status id
                ws.task.deleteStatus(taskStatusId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getStatus

Description:

Method	Return values	Description
getStatus(long statusId)	TaskStatus	Get a task status by id.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskId = 1; // Valid task id
                TaskStatus taskStatus= ws.task.getStatus(taskId);
                System.Console.WriteLine(taskStatus.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getNotes

Description:

Method	Return values	Description
getNotes(long taskId)	List<TaskNote>	Returns a list of notes for a task.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskId = 1; // Valid task id
            }
        }
    }
}

```

```

        foreach (TaskNote tn in ws.task.getNotes(taskId))
        {
            System.Console.WriteLine(tn.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

createNote

Description:

Method	Return values	Description
createNote(long taskId, String text)	TaskNote	Create a note for a certain task.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskId = 1; // Valid task id
                TaskNote taskNote = ws.task.createNote(taskId, "Any note");
                System.Console.WriteLine(taskNote.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

updateNote

Description:

Method	Return values	Description
updateNote(long noteId, String text)	TaskNote	Update a task note.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskNoteId = 1; // Valid task note id
                TaskNote taskNote = ws.task.updateNote(taskNoteId, "Any text");
                System.Console.WriteLine(taskNote.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

deleteNote

Description:

Method	Return values	Description
deleteNote(long noteId)	void	Delete a task note.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskNoteId = 1; // Valid task note id
                ws.task.deleteNote(taskNoteId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getHistory

Description:

Method	Return values	Description
getHistory(long taskId)	List<TaskHistory>	Returns a list for a task.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long taskId = 1; // Valid task id
            }
        }
    }
}

```

```

        foreach (TaskHistory th in ws.task.getHistory(taskId))
        {
            System.Console.WriteLine(th.toString());
        }
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

getRelatedTasks

Description:

Method	Return values	Description
getRelatedTasks(string uuid)	TaskList	Returns a list of tasks.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (TaskList tl in ws.task.getRelatedTasks("l8fce7e5-f667-44bd-be2f-f3b3e584d574").tasks)
                {
                    System.Console.WriteLine(tl.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

UserConfig samples

Basics

Suggested code sample

First, you must create the web service object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in to the web services, the session is kept in the web service object. Then you can use the other API methods.

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

```
ws.userConfig.getConfig();
```

Methods

getConfig

Description:

Method	Return values	Description
getConfig()	UserConfig	Returns the user settings.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
    }
```

```

static void Main(string[] args)
{
    String host = "http://localhost:8180/openkm";
    String username = "okmAdmin";
    String password = "admin";
    OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

    try
    {
        ws.login(user, password);
        UserConfig userCfg = ws.userConfig.getConfig();
        Console.WriteLine("User: " + userCfg.user);
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}

```

setHome

Description:

Method	Return values	Description
setHome(String uuid)	void	Sets the user's home node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.userConfig.setHome("53751cb7-c462-4320-ba46-4cc33e4667ae");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}
```

Wizard samples

Basics



Example of UUID:

- Using UUID -> "373bcdd0-c082-4e7b-addd-e10ef813946e";

Suggested code sample

First, you must create the webservice object:

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

Then you should log in using the method "**login**". You can access the "**login**" method from the web service object "**ws**" as shown below:

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



Once you are logged in with the web services, the session is kept in the web service object. Then you can use the other API methods.

At this point you can use all the Wizard methods from "**wizard**" class as shown below:

```
ws.wizard.findByUser();
```

Methods

findByUser

Description:

Method	Return values	Description
findByUser()	List<WizardNode>	Returns a list of nodes with a wizard.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;
```

```

using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<WizardNode> list = ws.wizard.findByUser();
                foreach (WizardNode wizardNode in list)
                {
                    Console.WriteLine(wizardNode.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

findAllByUser

Description:

Method	Return values	Description
findAllByUser()	List<WizardNode>	Returns a list of nodes with a wizard for the user.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";

```

```

String username = "okmAdmin";
String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    List<WizardNode> list = ws.wizard.findAllByUser();
    foreach (WizardNode wizardNode in list)
    {
        Console.WriteLine(wizardNode.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

findByUserAndUuid

Description:

Method	Return values	Description
findByUserAndUuid(String uuid)	WizardNode	Get the wizard of a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                WizardNode wn= ws.wizard.findByUserAndUuid("5458e52a-58a5-4b5d-ac89-e4e5837924a9");
                Console.WriteLine(wn.toString());
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```
        System.Console.WriteLine(e.ToString());
    }
}
```

hasWizardByUserAndNode

Description:

Method	Return values	Description
hasWizardByUserAndNode(String uuid)	bool	Returns whether a node has a wizard.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Console.WriteLine("Has a wizard active: " + ws.wizard.hasWizardByUserAndNode("5458e52a-58a5-4b5d"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

deleteAll

Description:

Method	Return values	Description

deleteAll()	void	Removes all wizards for the user.
--------------------	-------------	-----------------------------------

 Only wizards assigned to the user session will be removed.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.deleteAll();
                Console.WriteLine("Delete all");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

postponeNode

Description:

Method	Return values	Description
postponeNode(String uuid, DateTime date)	void	Postpone a node.

Example:

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DateTime date = DateTime.Now;
                ws.wizard.postponeNode("5458e52a-58a5-4b5d-ac89-e4e5837924a9", date);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

cancelPostponedNode

Description:

Method	Return values	Description
cancelPostponedNode(String uuid)	void	Cancel the postponed node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {

```

```

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

try
{
    ws.login(user, password);
    ws.wizard.cancelPostponedNode("5458e52a-58a5-4b5d-ac89-e4e5837924a9");
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

addDigitalSignature

Description:

Method	Return values	Description
addDigitalSignature(String uuid, int order)	void	Add a digital signature in the wizard of a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addDigitalSignature("055b5206-35d0-4351-ba92-c304bbba7ddf",order);
                Console.WriteLine("Add digital signature");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}
```

removeDigitalSignature

Description:

Method	Return values	Description
removeDigitalSignature(String uuid)	void	Remove a digital signature in the wizard of a node.

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.util;  
using com.openkm.sdk4csharp.bean;  
using com.openkm.sdk4csharp.impl;  
using System.IO;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                ws.wizard.removeDigitalSignature("055b5206-35d0-4351-ba92-c304bbba7ddf");  
                Console.WriteLine("Remove digital signature");  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

addShowWizardCategories

Description:

Method	Return values	Description

addShowWizardCategories(String uuid, int order)	void	Add show wizard categories to the wizard of a node.
--	-------------	---

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addShowWizardCategories("055b5206-35d0-4351-ba92-c304bbba7ddf", order);
                Console.WriteLine("addShowWizardCategories");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

removeShowWizardCategories

Description:

Method	Return values	Description
removeShowWizardCategories(String uuid)	void	Remove show wizard categories from the wizard of a node.

Example:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeShowWizardCategories("055b5206-35d0-4351-ba92-c304bbba7ddf");
                Console.WriteLine("removeShowWizardCategories");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

addShowWizardKeywords

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {

```

```
static void Main(string[] args)
{
    String host = "http://localhost:8080/openkm";
    String username = "okmAdmin";
    String password = "admin";
    OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

    try
    {
        ws.login(user, password);
        int order = 0;
        ws.wizard.addShowWizardKeywords("055b5206-35d0-4351-ba92-c304bbba7ddf", order);
        Console.WriteLine("addShowWizardKeywords");
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
```

removeShowWizardKeywords

Description:

Method	Return values	Description
removeShowWizardKeywords(String uuid)	void	Remove show wizard keywords from the wizard of a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeShowWizardKeywords("055b5206-35d0-4351-ba92-c304bbba7ddf");
            }
        }
    }
}
```

```

        Console.WriteLine("removeShowWizardKeywords");
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

addShowWizardOCRDataCapture

Description:

Method	Return values	Description
addShowWizardOCRDataCapture(String uuid, int order)	void	Add show wizard OCR data capture to the wizard of a node.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addShowWizardOCRDataCapture("055b5206-35d0-4351-ba92-c304bbba7ddf", order);
                Console.WriteLine("addShowWizardOCRDataCapture");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

removeShowWizardOCRDataCapture

Description:

Method	Return values	Description
removeShowWizardOCRDataCapture(String uuid)	void	Remove show wizard OCR data capture from the wizard of a node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeShowWizardOCRDataCapture("055b5206-35d0-4351-ba92-c304bbba7ddf");
                Console.WriteLine("removeShowWizardOCRDataCapture");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

addGroup

Description:

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



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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addGroup("055b5206-35d0-4351-ba92-c304bbba7ddf", "okg:testing", order);
                Console.WriteLine("addGroup");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

removeGroup

Description:

Method	Return values	Description
removeGroup(String uuid, String group)	void	Remove a metadata group from the wizard.



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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeGroup("055b5206-35d0-4351-ba92-c304bbba7ddf", "okg:testing");
                Console.WriteLine("removeGroup");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

addWorkflow

Description:

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



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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addWorkflow("055b5206-35d0-4351-ba92-c304bbba7ddf", "purchase", order);
                Console.WriteLine("addWorkflow");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

removeWorkflow

Description:

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



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

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest

```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeWorkflow("055b5206-35d0-4351-ba92-c304bbba7ddf", "purchase");
                Console.WriteLine("removeWorkflow");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

setAutostart

Description:

Method	Return values	Description
setAutostart(String uuid)	void	Set auto-start for the wizard.
 Autostart should always be set as the last wizard option.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

```

```

        try
        {
            ws.login(user, password);
            ws.wizard.setAutostart("055b5206-35d0-4351-ba92-c304bbba7ddf");
            Console.WriteLine("setAutostart");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

deleteByNode

Description:

Method	Return values	Description
deleteByNode(String uuid)	void	Delete a node in the wizard.

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

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.deleteByNode("055b5206-35d0-4351-ba92-c304bbba7ddf");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

```
}  
}  
}
```

Workflow samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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



Once you are logged in with the web services, the session is kept in the web service object. Then you can use the other API methods.

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

```
ws.workflow.registerProcessDefinition(is);
```

For most examples, the [Purchase workflow sample](#) has been used.

Methods

registerProcessDefinition

Description:

Method	Return values	Description
registerProcessDefinition(FileStream fs)	void	Registers a new workflow.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
```

```

public class Program
{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\Purchase.par", FileMode.Open);
            ws.workflow.registerProcessDefinition(fs);
            fs.Dispose();
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

deleteProcessDefinition

Description:

Method	Return values	Description
deleteProcessDefinition(long pdId)	void	Deletes a workflow.
The parameter pdId value is a valid workflow process definition ID.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 5; // Valid workflow process definition
                ws.workflow.deleteProcessDefinition(pdId);
            }
        }
    }
}

```

```
    }  
    catch (Exception e)  
    {  
        System.Console.WriteLine(e.ToString());  
    }  
}  
}
```

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 ID.		

Example:

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;  
using com.openkm.sdk4csharp;  
using com.openkm.sdk4csharp.bean;  
  
namespace OKMRest  
{  
    public class Program  
    {  
        static void Main(string[] args)  
        {  
            String host = "http://localhost:8080/openkm";  
            String username = "okmAdmin";  
            String password = "admin";  
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);  
  
            try  
            {  
                ws.login(user, password);  
                long pdId = 1; // Valid workflow process definition  
                ProcessDefinition pd = ws.workflow.getProcessDefinition(pdId);  
                System.Console.WriteLine(pd);  
            }  
            catch (Exception e)  
            {  
                System.Console.WriteLine(e.ToString());  
            }  
        }  
    }  
}
```

runProcessDefinition

Description:

Method	Return values	Description
runProcessDefinition(long pdId, String uuid, List<FormElement> values)	ProcessInstance	Executes a workflow on some node.
The parameter pdId value is a valid workflow process definition ID. The parameter uuid can be any document, mail, folder or record UUID. The parameter values are form element values needed to start the workflow (not all workflows require form values to start).		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 8041; // Some valid workflow process definition id
                List<FormElement> feList = new List<FormElement>();
                Input price = new Input();
                price.name = "price";
                price.value = "1000";
                feList.Add(price);
                TextArea textArea = new TextArea();
                textArea.name = "description";
                textArea.value = "some description here";
                feList.Add(textArea);
                ws.workflow.runProcessDefinition(pdId, "7aa523e0-06cf-4733-b8c8-cc74d8b2716d", feList);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

runProcessDefinition

Description:

Method	Return values	Description
runProcessDefinition(long pdId, Dictionary<String, String> properties)	ProcessInstance	Executes a workflow on some node.
The parameter pdId value is a valid workflow process definition ID. The parameter values are form element values needed to start the workflow (not all workflows require form values to start).		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 8041; // Some valid workflow process definition id
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("price", "1000");
                properties.Add("description", "some description here");
                ws.workflow.runProcessDefinition(pdId, properties);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findAllProcessDefinitions

Description:

Method	Return values	Description

findAllProcessDefinitions()	List<ProcessDefinition>	Retrieves a list of all registered workflow definitions.
------------------------------------	--------------------------------------	--

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (ProcessDefinition pd in ws.workflow.findAllProcessDefinitions())
                {
                    System.Console.WriteLine(pd);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findProcessInstances

Description:

Method	Return values	Description
findProcessInstances(long pdId)	List<ProcessInstance>	Retrieves a list of all process instances of some registered workflow definition.
The parameter pdId value is a valid workflow process definition.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```

using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Get all workflow definitions
                foreach (ProcessDefinition pd in ws.findAllProcessDefinitions())
                {
                    System.Console.WriteLine("WF definition: "+pd);
                    // Get all process of some workflow definition
                    foreach (ProcessInstance pi in ws.workflow.findProcessInstances(pd.id))
                    {
                        System.Console.WriteLine("PI: "+pi);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

findLatestProcessDefinitions

Description:

Method	Return values	Description
findLatestProcessDefinitions()	List<ProcessDefinition>	Retrieves a list of the latest workflow definitions.
 There can be several versions of the same workflow registered.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest

```

```

{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Get all latest workflow definitions
                foreach (ProcessDefinition pd in ws.workflow.findLatestProcessDefinitions())
                {
                    System.Console.WriteLine("WF definition: " + pd);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

findLastProcessDefinition

Description:

Method	Return values	Description
findLastProcessDefinition(String name)	ProcessDefinition	Retrieves the latest workflow definition of a specific workflow.
The parameter name identifies a specific group of workflow definitions.  Several workflow definition versions can have the same name.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)

```

```

    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            ProcessDefinition pd = ws.workflow.findLastProcessDefinition("purchase");
            System.Console.WriteLine("WF definition: " + pd);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

getProcessInstance

Description:

Method	Return values	Description
getProcessInstance(long piId)	ProcessInstance	Returns the process instance.
The parameter piId is a valid process instance ID.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long piId = 5868; // Some valid process instance id
                ProcessInstance pi = ws.workflow.getProcessInstance(piId);
                System.Console.WriteLine("PI: " + pi);
            }
            catch (Exception e)
            {
            }
        }
    }
}

```

```

        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

findUserTaskInstances

Description:

Method	Return values	Description
findUserTaskInstances(int offset, int limit)	TaskInstanceResultSet	Returns a TaskInstanceResultSet object.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

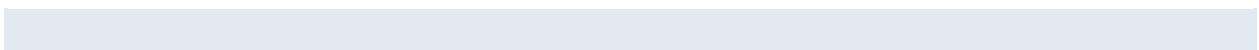
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Get all user task instances
                TaskInstanceResultSet tasks = ws.workflow.findUserTaskInstances(0, 5);
                foreach (TaskInstance ti in tasks.results)
                {
                    System.Console.WriteLine(ti.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

findTaskInstances

Description:



Method	Return values	Description
findTaskInstances(long pId)	List<TaskInstance>	Retrieves a list of task instances for a given process instance ID.
The parameter pId is a valid process instance id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Get all task instances of some process instance
                long pId = 8108; // Some valid process instance id
                foreach (TaskInstance ti in ws.workflow.findTaskInstances(pId))
                {
                    System.Console.WriteLine(ti);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

setTaskInstanceValues

Description:

Method	Return values	Description
setTaskInstanceValues(long tiId, String transName, List<FormElement> values)	void	Retrieves a list of task instances of some process instance ID.

The parameter tiId is a valid task instance id.

The parameter transName is the chosen transaction.

The parameter values are form element values needed for starting the workflow (not all workflow tasks require form values).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long tiId = 8110; // Some valid task instance id
                List<FormElement> feList = new List<FormElement>();
                ws.workflow.setTaskInstanceValues(tiId, "approve", feList);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getTaskInstance

Description:

Method	Return values	Description
getTaskInstance(long tiId)	TaskInstance	Returns a task instance.
The parameter tiId is a valid task instance id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long tild = 8110; // Some valid task instance id
                TaskInstance ti = ws.workflow.getTaskInstance(tild);
                System.Console.WriteLine(ti);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

startTaskInstance

Description:

Method	Return values	Description
startTaskInstance(long tiId)	void	Starts a task instance.
The parameter tiId is a valid task instance id.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
```

```

{
    static void Main(string[] args)
    {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

        try
        {
            ws.login(user, password);
            long tild = 8110; // Some valid task instance id
            ws.workflow.startTaskInstance(tild);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
}

```

setTaskInstanceActorId

Description:

Method	Return values	Description
setTaskInstanceActorId(long tild, String actorId)	void	Starts a task instance.
The parameter tild is a valid task instance id. The parameter actorId must be a valid OpenKM userId.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long tild = 8110; // Some valid task instance id
            }
        }
    }
}

```

```

        ws.workflow.setTaskInstanceActorId(tild, "okmAdmin");
    }
    catch (Exception e)
    {
        System.Console.WriteLine(e.ToString());
    }
}
}
}

```

endTaskInstance

Description:

Method	Return values	Description
endTaskInstance(long tiId, String transName)	void	Starts a task instance.
The parameter tiId is a valid task instance id. The parameter transName is a transaction name.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long tild = 8110; // Some valid task instance id
                ws.workflow.endTaskInstance(tild, "end");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

getProcessDefinitionForms

Description:

Method	Return values	Description
getProcessDefinitionForms(long pdId)	Dictionary<String, List<FormElement>>	Returns a map with all the process definition forms.
The parameter pdId value is a valid workflow process definition ID.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, List<FormElement>> forms = ws.workflow.getProcessDefinitionForms(1);
                foreach (KeyValuePair<String, List<FormElement>> pair in forms)
                {
                    System.Console.WriteLine("Key:" + pair.Key);
                    foreach (FormElement fe in pair.Value)
                    {
                        System.Console.WriteLine("Fe:" + fe);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

getProcessDefinitionImage

Description:

Method	Return values	Description

getProcessDefinitionImage(string pdId, string uuid)	string	Returns an image as a string.
The parameter pdId value is a valid workflow process definition ID.		

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                string tild = "1";
                string res = ws.workflow.getProcessDefinitionImage(tild, "a978f8e6-aa87-4f0f-b7bc-6f44cb090fdf");
                System.Console.WriteLine(res);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

findPooledTaskInstances

Description:

Method	Return values	Description
findPooledTaskInstances()	TaskInstanceResultSet	Returns a TaskInstanceResultSet object.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                TaskInstanceResultSet tasks = ws.workflow.findPooledTaskInstances();
                foreach (TaskInstance ti in tasks.results)
                {
                    System.Console.WriteLine(ti.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

findProcessInstancesByNode

Description:

Method	Return values	Description
findProcessInstancesByNode(String uuid)	List<ProcessInstance>	Retrieves a list of all process instances by node for some registered workflow definition.
The parameter uuid can be any document, mail, folder or record UUID.		

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";

```

```

String password = "admin";
OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

try
{
    ws.login(user, password);
    string uuid = "926ee2b3-44d8-4f8a-8b0a-864a0d829971";
    foreach (ProcessInstance pi in ws.workflow.findProcessInstancesByNode(uuid))
    {
        System.Console.WriteLine("PI: " + pi.toString());
    }
}
catch (Exception e)
{
    System.Console.WriteLine(e.ToString());
}
}
}

```

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:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 2;
                String taskName = "task-elaboration";
                Console.WriteLine(ws.workflow.getSuggestBoxKeyValue(pdId, "9c069bdc-4654-4066-b4b2-d203b23e58f
            }
            catch (Exception e)
            {

```

```

        System.Console.WriteLine(e.ToString());
    }
}
}

```

getSuggestBoxKeyValuesFiltered

Description:

Method	Return values	Description
getSuggestBoxKeyValuesFiltered(long pdId, String uuid, String taskName, String propertyName, String filter)	Dictionary<String, String>	Retrieves a map of (key, value) pairs with suggestBox values.

Example:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 2;
                Dictionary<String, String> values = ws.workflow.getSuggestBoxKeyValuesFiltered(pdId, "9c069bdc-4654-40a0-b060-000000000000", "taskName", "propertyName", "filter");
                foreach (KeyValuePair<string, string> value in values)
                {
                    Console.WriteLine(value.Key + ":" + value.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}

```

searchAssignedTasks

Description:

Method	Return values	Description
searchAssignedTasks(DateTime? start, DateTime? end, String status, long procDefId, int offset, int limit)	ProcessInstanceResultSet	Return workflow tasks assigned to the logged user with date range, status, process definition, and pagination parameter.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 2;
                DateTime startDate = DateTime.Now.AddMonths(-2);
                DateTime endDate = DateTime.Now;
                ProcessInstanceResultSet assignedTask = ws.workflow.searchAssignedTasks(startDate, endDate, "runni
                foreach(var result in assignedTask.results)
                {
                    Console.WriteLine(result.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

searchStartedByMe

Description:

Method	Return values	Description

searchStartedByMe(DateTime? start, DateTime? end, String status, long procDefId, int offset, int limit)	ProcessInstanceResultSet	Return workflow tasks started by the logged user with date range, status, process definition, and pagination parameter.
--	---------------------------------	---

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long pdId = 2;
                DateTime startDate = DateTime.Now.AddMonths(-2);
                DateTime endDate = DateTime.Now;
                ProcessInstanceResultSet startedByMe= ws.workflow.searchStartedByMe(startDate, endDate, "running",
                foreach(var result in startedByMe.results)
                {
                    Console.WriteLine(result.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

searchOverview

Description:

Method	Return values	Description
searchOverview(DateTime? start, DateTime? end, String status, long procDefId, String initiator,	ProcessInstanceResultSet	Return a general overview of all users' workflow tasks with date range, status, process definition, initiator, task user, and

String taskUser, int offset, int limit)		pagination parameters.
--	--	------------------------

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

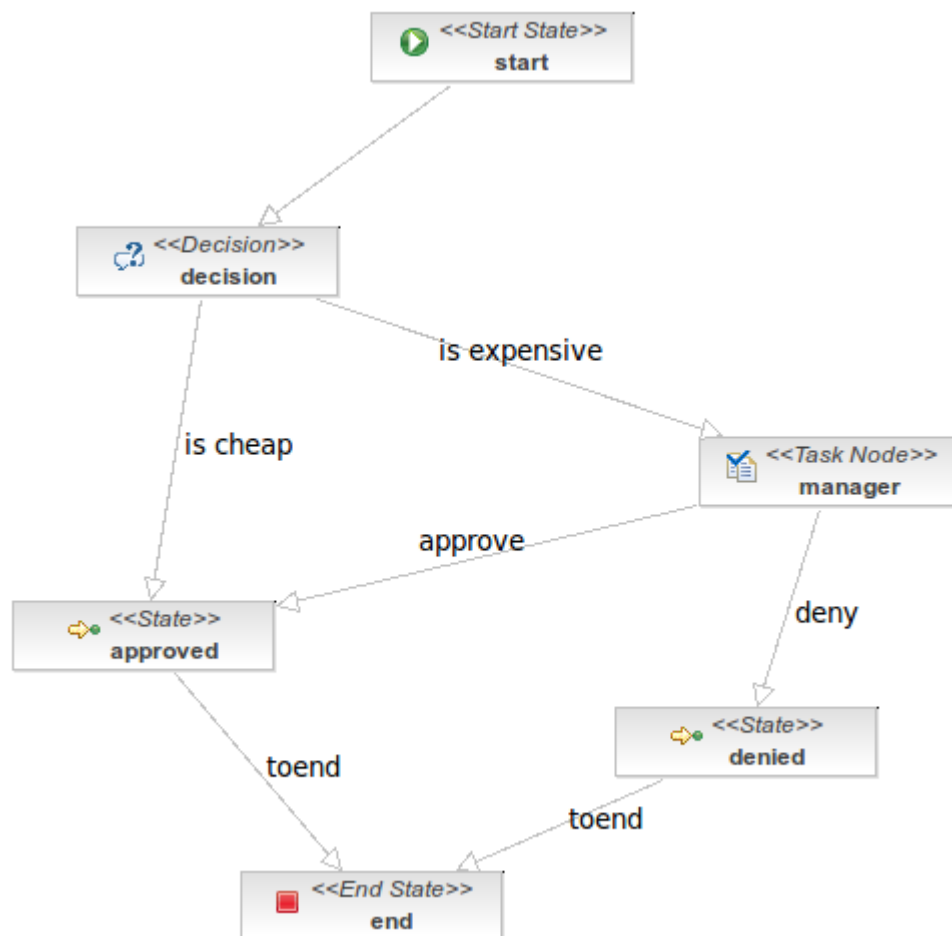
            try
            {
                ws.login(user, password);
                long pdId = 2;
                DateTime startDate = DateTime.Now.AddMonths(-2);
                DateTime endDate = DateTime.Now;
                String initiator = "requests_user";
                String taskUser = "approver_user";
                ProcessInstanceResultSet overview = ws.workflow.searchOverview(startDate, endDate, "running", pdDefl
                foreach(var result in overview.results)
                {
                    Console.WriteLine(result.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

Purchase workflow sample

The task will be assigned to a user called "manager", so you need to create this user and log in as him 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 <= 500}"></condition>
    </transition>
  </decision>

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

  <task name="manager">
    <transition to="approved" name="approve"></transition>
    <transition to="denied" name="deny"></transition>
  </task>

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

  <end-state name="end"></end-state>
</process-definition>
  
```

```

    </transition>
    <transition to="manager" name="is expensive">
      <condition expression="#{price.value > 500}"></condition>
    </transition>
  </decision>

  <task-node name="manager">
    <task name="evaluate price">
      <description>The manager may deny purchase or go ahead.</description>
      <assignment actor-id="manager"></assignment>
    </task>
    <transition to="denied" name="deny"></transition>
    <transition to="approved" name="approve"></transition>
  </task-node>

  <state name="approved">
    <description>The purchase has been approved.</description>
    <timer duedate="15 seconds" name="approved timer" transition="toend">
      <script>print("&quot;From APPROVED Go to END&quot;");</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <state name="denied">
    <description>The purchase has been denied.</description>
    <timer duedate="15 seconds" name="denied timer" transition="toend">
      <script>print("&quot;From DENIED Go to END&quot;");</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <end-state name="end"></end-state>
</process-definition>

```

Process handlers

None.

Form definition

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
"http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="run_config">
    <input label="Purchase price" name="price" />
    <textarea label="Purchase description" name="description" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="evaluate price">
    <input label="Purchase price" name="price" data="price" readonly="true" />
    <textarea label="Purchase description" name="description" data="description" readonly="true" />
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>

```