



Documentation for SDK for PHP 2.6.2

Table of Contents

Table of Contents	2
SDK for PHP 2.6.2	7
License	7
Compatibility	7
Download	7
Sample client	8
Basic concepts	9
Authentication	9
Accessing API	11
Class Hierarchy	13
Auth samples	16
Basics	16
Methods	16
login	16
getGrantedRoles	17
getGrantedUsers	18
getRoles	19
getUsers	20
grantRole	21
grantUser	22
revokeRole	23
revokeUser	24
getRolesByUser	25
getUsersByRole	26
getMail	27
getName	28
createUser	29
deleteUser	30
updateUser	31
createRole	32
deleteRole	33
updateRole	33
assignRole	34
removeRole	35
changeSecurity	36
getProfiles	37
getUserProfile	38
setUserProfile	39
Conversion samples	41
Methods	41
doc2pdf	41
pdf2swf	42
imageConvert	43
html2pdf	45
doc2txt	46
img2txt	47
barcode2txt	48
Document samples	50
Basics	50
Methods	50
createDocumentSimple	50
deleteDocument	51
getDocumentProperties	52
getContent	53
getContentByVersion	54
getExtractedText	55
getThumbnail	56
getDocumentChildren	57

renameDocument	58
setProperties	59
setLanguage	61
setDocumentTitle	61
createFromTemplate	62
checkout	62
cancelCheckout	63
forceCancelCheckout	64
isCheckedOut	65
checkin	66
getVersionHistory	67
lock	68
unlock	69
forceUnlock	70
isLocked	71
getLockInfo	72
purgeDocument	73
moveDocument	74
copyDocument	75
restoreVersion	76
purgeVersionHistory	77
getVersionHistorySize	78
isValidDocument	79
getDocumentPath	80
getDetectedLanguages	81
extendedDocumentCopy	81
createFromTemplate	83
createFromTemplateSimple	85
updateFromTemplate	87
updateFromTemplateSimple	89
getAnnotations	90
getDifferences	91
getCheckedOut	92
Folder samples	94
Basics	94
Methods	94
createFolder	94
createFolderSimple	95
getFolderProperties	96
deleteFolder	97
renameFolder	98
moveFolder	99
getFolderChildren	99
isValidFolder	100
getFolderPath	101
copyFolder	102
extendedFolderCopy	103
getContentInfo	105
purgeFolder	106
setStyle	107
createMissingFolders	108
Mail samples	110
Basics	110
Methods	110
createMail	110
getMailProperties	112
deleteMail	113
renameMail	114
moveMail	115
getMailChildren	116
isValidMail	117
getMailPath	118
createAttachment	118
deleteAttachment	119
getAttachments	120
purgeMail	121
copyMail	122

extendedMailCopy	123
sendMailWithAttachments	125
importEml	126
importMsg	127
setMailTitle	128
sendMail	129
Note samples	131
Basics	131
Methods	131
addNote	131
getNote	132
deleteNote	133
setNote	134
listNotes	135
Property samples	137
Basics	137
Methods	137
addCategory	137
removeCategory	138
addKeyword	139
removeKeyword	140
setEncryption	141
unsetEncryption	142
setSigned	143
PropertyGroup samples	145
Basics	145
Methods	145
addGroup	145
removeGroup	146
getGroups	147
getAllGroups	148
getPropertyGroupProperties	149
getPropertyGroupPropertiesSimple	150
getPropertyGroupForm	151
setPropertyGroupProperties	152
setPropertyGroupPropertiesSimple	153
hasGroup	154
getRegisteredDefinition	155
getSuggestions	156
registerDefinition	158
Record samples	160
Methods	160
createRecord	160
getRecordProperties	161
deleteRecord	162
purgeRecord	163
renameRecord	164
moveRecord	165
copyRecord	166
isValidRecord	167
getRecordChildren	167
lockRecord	168
unlockRecord	169
forceUnlockRecord	170
setRecordTitle	171
getRecordPath	172
Relation samples	174
Basics	174
Methods	174
getRelationTypes	174
addRelation	175
deleteRelation	176
getRelations	177
getRelationGroups	178

addRelationGroup	179
addNodeToGroup	180
deleteRelationGroup	181
findRelationGroup	182
setRelationGroupName	183
Repository samples	185
Methods	185
getRootFolder	185
getTrashFolder	185
getTrashFolderBase	186
getTemplatesFolder	187
getPersonalFolder	188
getPersonalFolderBase	189
getMailFolder	189
getMailFolderBase	190
getThesaurusFolder	191
getCategoriesFolder	192
purgeTrash	193
getUpdateMessage	194
getRepositoryUuid	195
hasNode	195
getNodePath	196
getNodeUuid	197
getAppVersion	198
copyAttributes	199
executeScript	200
executeSqlQuery	201
executeHqlQuery	203
getTranslations	204
getConfiguration	205
getChangeLog	206
Search samples	209
Basics	209
Methods	212
findByContent	212
findByName	213
findByKeywords	214
find	215
findPaginated	216
findByQuery	217
findByQueryPaginated	219
findSimpleQueryPaginated	220
findMoreLikeThis	222
getKeywordMap	223
getCategorizedDocuments	224
saveSearch	225
updateSearch	226
getSearch	227
getAllSearchs	228
deleteSearch	229
Workflow samples	231
Methods	231
registerProcessDefinition	231
deleteProcessDefinition	232
getProcessDefinition	233
runProcessDefinition	233
FindAllProcessDefinitions	235
findProcessInstances	236
findLatestProcessDefinitions	237
findLastProcessDefinition	238
getProcessInstance	238
findUserTaskInstances	239
findTaskInstances	240
setTaskInstanceValues	241
getTaskInstance	242
setTaskInstanceActorId	243

startTaskInstance	244
endTaskInstance	245
getProcessDefinitionForms	246
Purchase workflow sample	248
Process image	248
Process definition	248
Process handlers	249
Form definition	249
Report samples	250
Methods	250
getReports	250
getReport	251
executeReport	251

SDK for PHP 2.6.2

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

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

License



SDK for PHP 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 but 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 PHP version 2.6.2 should be used from OpenKM version 6.4.45 and upper.

Download

Download professional resources from OpenKM [Download center](#).

Sample client

Your first class:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class TestOKM {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function test() {
        $folders = $this->ws->getFolderChildren("/okm:root/SDK4PHP");
        foreach ($folders as $folder) {
            var_dump($folder);
        }
    }

}

$openkm = new OpenKM(); //autoload
$testOKM = new TestOKM();
$testOKM->test();
?>
```

Basic concepts

Authentication

The first lines in your PHP code should be used to create the Webservices object.

We suggest using this method:

```
$this->ws = OKMWebServicesFactory::build(host, user, password);
```

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;
use Httpful\Exception\ConnectionErrorException;
use openkm\exception\AccessDeniedException;
use openkm\exception\PathNotFoundException;
use openkm\exception\RepositoryException;
use openkm\exception\DatabaseException;
use openkm\exception\UnknowException;

class Example {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetAppVersion() {
        try {
            $appVersion = $this->ws->getAppVersion();
            var_dump($appVersion);
        } catch (AccessDeniedException $ade) {
            var_dump($ade);
        } catch (PathNotFoundException $pnfe) {
            var_dump($pnfe);
        } catch (RepositoryException $re) {
            var_dump($re);
        } catch (DatabaseException $de) {
            var_dump($de);
        } catch (UnknowException $ue) {
            var_dump($ue);
        } catch (ConnectionErrorException $cee) {
            var_dump($cee);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$example = new Example();
```

```
$example->testGetAppVersion();  
?>
```

Also is possible doing the same from each API class implementation.



We do not suggest this way.

For example with this method:

```
$this->repository = new RepositoryImpl(self::HOST, self::USER, self::PASSWORD);
```

```
<?php  
include '../src/openkm/OpenKM.php';  
  
use openkm\impl\RepositoryImpl;  
use openkm\OpenKM;  
use openkm\bean\AppVersion;  
use Httpful\Exception\ConnectionErrorException;  
use openkm\exception\AccessDeniedException;  
use openkm\exception\PathNotFoundException;  
use openkm\exception\RepositoryException;  
use openkm\exception\DatabaseException;  
use openkm\exception\UnknowException;  
  
class Test {  
  
    const HOST = "http://localhost:8080/OpenKM/";  
    const USER = "okmAdmin";  
    const PASSWORD = "admin";  
  
    private $repository;  
  
    public function __construct() {  
        $this->repository = new RepositoryImpl(self::HOST, self::USER, self::PASSWORD);  
    }  
  
    public function testGetAppVersion() {  
        try {  
            $appVersion = $this->repository->getAppVersion();  
            var_dump($appVersion);  
        } catch (AccessDeniedException $ade) {  
            var_dump($ade);  
        } catch (PathNotFoundException $pnfe) {  
            var_dump($pnfe);  
        } catch (RepositoryException $re) {  
            var_dump($re);  
        } catch (DatabaseException $de) {  
            var_dump($de);  
        } catch (UnknowException $ue) {  
            var_dump($ue);  
        } catch (ConnectionErrorException $cee) {  
            var_dump($cee);  
        } catch (Exception $e) {  
            var_dump($e);  
        }  
    }  
  
}
```

```

$openkm = new OpenKM(); //autoload
$test = new Test();
$test->testGetAppVersion();

?>

```

Accessing API

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



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

At the moment of writing this page the actual OpenKM version was 6.4.22 what can change on time.



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

OpenKM API classes:

OpenKM API class	Description	Supported	Implementation	Interface
OKMAuth	Manages security and users. For example adds or removes grants on a node, creates or modifies users or getting the profiles.	Yes	AuthImpl.php	BaseAuth.php
OKMBookmark	Manages the user's bookmarks.	No		
OKMDashboard	Manages all data shown at the dashboard.	No		
OKMDocument	Manages documents. For example creates, moves or deletes a document.	Yes	DocumentImpl.php	BaseDocument.php
OKMFolder	Manages folders. For	Yes	FolderImpl.php	BaseFolder.php

	example creates, moves or deletes a folder.			
OKMMail	Manages mails. For example creates, moves or deletes a mail.	Yes	MailImpl.php	BaseMail.php
OKMNote	Manages notes on any node type. For example creates, edits or deletes a note on a document, folder, mail or record.	Yes	NoteImpl.php	BaseNote.php
OKMNotification	Manages notifications. For example adds or removes subscriptions on a document or a folder.	No	NotificationImpl.php	BaseNotification.php
OKMProperty	Manages categories and keywords. For example adds or removes keywords on a document, folder, mail or record.	Yes	PropertyImpl.php	BaseProperty.php
OKMPropertyGroup	Manages metadata. For example add metadata group, set metadata fields.	Yes	PropertyGroupImpl.php	BasePropertyGroup.php
OKMRecord	Manages records. For example creates, moves or deletes a record.	Yes	RecordImpl.php	BaseRecord.php

OKMRelation	Manages relations between nodes. For example creates a relation (parent-child) between two documents.	Yes	RelationImpl.php	BaseRelation.php
OKMRepository	A lot of stuff related with repository. For example gets the properties of main root node (/okm:root).	Yes	RepositoryImpl.php	BaseRepository.php
OKMSearch	Manages search feature. For example manages saved queries or perform a new query to the repository.	Yes	SearchImpl.php	BaseSearch.php
OKMStats	General stats of the repository.	No		
OKMTask	Manages task. For example creates a new task.	No		
OKMUserConfig	Manages user home configuration.	No		
OKMWorkflow	Manages workflows. For example executes a new workflow.	Yes	WorkflowImpl.php	BaseWorkflow.php

Class Hierarchy

Packages detail:

Name	Description
com.openkm	<p>The openkm.OKMWebservicesFactory that returns an openkm.OKMWebservices object which implements all interfaces.</p> <pre data-bbox="424 501 1401 555">\$this->ws = OKMWebServicesFactory::build(host, user, password);</pre>
openkm.bean	Contains all classes result of unmarshalling REST objects.
openkm.definition	<p>All interface classes:</p> <ul data-bbox="485 792 884 1375" style="list-style-type: none"> • openkm.definition.BaseAuth • openkm.definition.BaseDocument • openkm.definition.BaseFolder • openkm.definition.BaseMail • openkm.definition.BaseNote • openkm.definition.BaseProperty • openkm.definition.BasePropertyGroup • openkm.definition.BaseRecord • openkm.definition.BaseRelation • openkm.definition.BaseRepository • openkm.definition.BaseSearch • openkm.definition.BaseWorkflow
openkm.impl	<p>All interface implementation classes:</p> <ul data-bbox="485 1536 836 1966" style="list-style-type: none"> • openkm.impl.AuthImpl • openkm.impl.DocumentImpl • openkm.impl.FolderImpl • openkm.impl.MailImpl • openkm.impl.NoteImpl • openkm.impl.PropertyGroupImpl • openkm.impl.PropertyImpl • openkm.impl.RecordImpl • openkm.impl.RelationImpl

	<ul style="list-style-type: none">• openkm.impl.RepositoryImpl• openkm.impl.SearchImpl• openkm.impl.WorkflowImpl
openkm.util	A couple of utilities.
openkm.exception	All exception classes.

Auth samples

Basics

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

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

```
$permission = Permission::READ + Permission::WRITE;
```

To check if you have permission access you should do:

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

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

 Example of `nodeId`:

- Using UUID -> "`96c44de6-1d0d-45fb-b380-4984f46bbeb3`";
- Using path -> "`/okm:root/SDK4PHP/sample.pdf`"

Methods

login

Description:

Method	Return values	Description
<code>login()</code>	<code>void</code>	Simulate first login process

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

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

Otherwise you can get errors, for example `PathNotExistsException /okm:trash/userId` when deleting objects,

because the folders has not been created.

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testLogin() {
        try {
            $this->ws->login();
            echo 'login';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testLogin();
?>
```

getGrantedRoles

Description:

Method	Return values	Description
getGrantedRoles(\$nodeId)	array	Return the granted roles of a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record.		

Example:

```
<?php
```

```

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetGrantedRoles() {
        try {
            $grantedRoles = $this->ws->getGrantedRoles('/okm:root/SDK4PHP');
            foreach ($grantedRoles as $role) {
                var_dump($role);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetGrantedRoles();
?>

```

getGrantedUsers

Description:

Method	Return values	Description
getGrantedUsers(\$nodeId)	array	Returns the granted users of a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record.		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

```

```

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetGrantedUsers() {
        try {
            $grantedUsers = $this->ws->getGrantedUsers('/okm:root/SDK4PHP');
            foreach ($grantedUsers as $user) {
                var_dump($user);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetGrantedUsers();
?>

```

getRoles

Description:

Method	Return values	Description
getRoles()	array	Returns the list of all the roles.

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRoles() {

```

```

        try {
            $roles = $this->ws->getRoles();
            foreach ($roles as $role) {
                var_dump($role);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetRoles();
?>

```

getUsers

Description:

Method	Return values	Description
getUsers()	array	Returns the list of all the users.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetUsers() {
        try {
            $users = $this->ws->getUsers();
            foreach ($users as $user) {
                var_dump($user);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload

```

```
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetUsers();
?>
```

grantRole

Description:

Method	Return values	Description
grantRole(\$nodeId, \$role, \$permissions, \$recursive)	void	Adds a role grant on a node.

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$role string type

\$permissions int type

\$recursive bool type

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

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGrantRole() {
        try {
            // Add ROLE_USER write grants at the node but not descendants
            $this->ws->grantRole("/okm:root/SDK4PHP", "ROLE_USER", openkm\bean\Permissions::ROLE_USER);

            // Add all ROLE_ADMIN grants to the node and descendants
            $this->ws->grantRole("/okm:root/SDK4PHP", "ROLE_ADMIN", openkm\bean\Permissions::ROLE_ADMIN);
        } catch (Exception $e) {
            // ...
        }
    }
}
```

```

        echo 'grant Role';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGrantRole();
?>

```

grantUser

Description:

Method	Return values	Description
grantUser(\$nodeId, \$user, \$permissions, \$recursive)	void	Adds a user grant on a node.

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$user string type

\$permissions int type

\$recursive bool type

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

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

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```

        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGrantUser() {
        try {
            // Add sochoa write grants at the node but not descendants
            $this->ws->grantUser("/okm:root/SDK4PHP", "sochoa", \openkm\bean\Permissions::WRITE);

            // Add all okmAdmin grants at the node and descendants
            $this->ws->grantUser("/okm:root/SDK4PHP", "okmAdmin", \openkm\bean\Permissions::ALL);

            echo 'grant User';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGrantUser();
?>

```

revokeRole

Description:

Method	Return values	Description
revokeRole(\$nodeId, \$role, \$permissions, \$recursive)	void	Removes a role grant on a node.

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$role string type

\$permissions int type

\$recursive bool type

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

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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

```

```

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRevokeRole() {
        try {
            // Remove ROLE_USER write grants at the node but not descendants
            $this->ws->revokeRole("/okm:root/SDK4PHP", "ROLE_USER", \openkm\bean\Permission);

            // Remove all ROLE_ADMIN grants to the node and descendants
            $this->ws->revokeRole("/okm:root/SDK4PHP", "ROLE_ADMIN", \openkm\bean\Permission);

            echo 'revoke Role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testRevokeRole();
?>

```

revokeUser

Description:

Method	Return values	Description
revokeUser(\$nodeId, \$user, \$permissions, \$recursive)	void	Removes a user grant on a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record.</p> <p>\$user string type</p> <p>\$permissions int type</p> <p>\$recursive bool type</p> <p>The parameter recursive only makes sense when the nodeId is a folder or record node.</p> <p>When parameter recursive is true, the change will be applied to the node and descendants.</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRevokeUser() {
        try {
            // Remove sochoa write grants at the node but not descendants
            $this->ws->revokeUser("/okm:root/SDK4PHP", "sochoa", \openkm\bean\Permissions);

            // Remove all okmAdmin grants at the node and descendants
            $this->ws->revokeUser("/okm:root/SDK4PHP", "okmAdmin", \openkm\bean\Permissions);
            echo 'revoke User';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testRevokeUser();
?>
```

getRolesByUser

Description:

Method	Return values	Description
getRolesByUser(\$user)	array	Returns the list of all the roles assigned to a user.
Parameters:		
\$user string type is the user		

Example:

```
<?php
```

```

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRolesByUser() {
        try {
            $roles = $this->ws->getRolesByUser('okmAdmin');
            foreach ($roles as $role) {
                var_dump($role);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetRolesByUser();
?>

```

getUsersByRole

Description:

Method	Return values	Description
getUsersByRole(\$role)	array	Returns the list of all the users who have assigned a role.
Parameters:		
\$role string type is the role		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

```

```

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetUsersByRole() {
        try {
            $users = $this->ws->getUsersByRole('ROLE_ADMIN');
            foreach ($users as $user) {
                var_dump($user);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetUsersByRole();
?>

```

getMail

Description:

Method	Return values	Description
getMail(\$user)	string	Returns the mail of a valid user.
Parameters:		
\$user string type is the user		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetMail() {
    try {
        var_dump($this->ws->getMail('okmAdmin'));
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetMail();
?>

```

getName

Description:

Method	Return values	Description
getName(\$user)	string	Returns the name of a valid user.
Parameters:		
\$user string type is the user		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetName() {
        try {
            var_dump($this->ws->getName('okmAdmin'));
        }
    }
}

```

```

        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetName();
?>

```

createUser

Description:

Method	Return values	Description
createUser(\$user, \$password, \$email, \$name, \$active)	void	Create a new user.
<p>Parameters:</p> <p>\$user string type</p> <p>\$password string type</p> <p>\$email string type</p> <p>\$name string type</p> <p>\$active bool type</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCreateUser() {
        try {
            $this->ws->createUser('gnu.java.sergio', "sochoa", "sochoa@openkm.com", "sochoa");
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

        echo 'create user';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testCreateUser();
?>

```

deleteUser

Description:

Method	Return values	Description
deleteUser(\$user)	void	Delete a user.
Parameters:		
\$user string type		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteUser() {
        try {
            $this->ws->deleteUser('gnu.java.sergio');
            echo 'delete user';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
}

```

```

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testDeleteUser();
?>

```

updateUser

Description:

Method	Return values	Description
updateUser(\$user, \$password, \$email, \$name, \$active)	void	Update a user.
<p>Parameters:</p> <p>\$user string type</p> <p>\$password string type</p> <p>\$email string type</p> <p>\$name string type</p> <p>\$active bool type</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUpdateUser() {
        try {
            $this->ws->updateUser("gnu.java.sergio", "sochoa", "sochoa@openkm.com", "sochoa");
            echo 'update user';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testUpdateUser();
?>

```

createRole

Description:

Method	Return values	Description
createRole(\$role, \$active)	void	Create a new role.
Parameters:		
\$role string type		
\$active bool type		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCreateRole() {
        try {
            $this->ws->createRole('ROLE_MANAGER', true);
            echo 'create role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload

```

```
$exampleAuth = new ExampleAuth();
$exampleAuth->testCreateRole();
?>
```

deleteRole

Description:

Method	Return values	Description
deleteRole(\$role)	void	Delete a role.
Parameters:		
\$role string type		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteRole() {
        try {
            $this->ws->deleteRole('ROLE_MANAGER');
            echo 'delete role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testDeleteRole();
?>
```

updateRole

Description:

Method	Return values	Description
updateRole(\$role, \$active)	void	Update a role.

Parameters:

\$role string type

\$active bool type

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUpdateRole() {
        try {
            $this->ws->updateRole('ROLE_MANAGER', true);
            echo 'update role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testUpdateRole();
?>
```

assignRole

Description:

Method	Return values	Description
assignRole(\$user, \$role)	void	Assign role to a user.
Parameters:		
\$user string type		
\$role string type		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testAssignRole() {
        try {
            $this->ws->assignRole('gnu.java.sergio', 'ROLE_MANAGER');
            echo 'assign role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testAssignRole();
?>
```

removeRole

Description:

Method	Return values	Description
removeRole(\$user, \$role)	void	Remove a role from a user.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRemoveRole() {
        try {
            $this->ws->removeRole('gnu.java.sergio', 'ROLE_MANAGER');
            echo 'remove role';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testRemoveRole();
?>
```

changeSecurity

Description:

Method	Return values	Description
changeSecurity(ChangeSecurity \$changeSecurity)	void	Change the security of a node.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
```

```

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testChangeSecurity() {
        try {
            $changeSecurity = new \openkm\bean\ChangeSecurity();
            $changeSecurity->setNodeId("768748c5-04ab-4921-b0d4-7545d239ce5a");
            $changeSecurity->setRecursive(true);

            $grantedRoles = array();
            $grantedRole = new \openkm\bean\GrantedRole();
            $grantedRole->setRole('ROLE_MANAGER');
            $grantedRole->setPermissions(\openkm\bean\Permission::READ + \openkm\bean\Permission::WRITE);
            $grantedRoles[] = $grantedRole;

            $changeSecurity->setGrantedRoles($grantedRoles);

            $grantedUsers = array();
            $grantedUser = new \openkm\bean\GrantedUser();
            $grantedUser->setUser('gnu.java.sergio');
            $grantedUser->setPermissions(\openkm\bean\Permission::READ + \openkm\bean\Permission::WRITE);
            $grantedUsers[] = $grantedUser;
            $changeSecurity->setGrantedUsers($grantedUsers);

            $revokedRoles = array();
            $revokedRole = new \openkm\bean\RevokedRole();
            $revokedRole->setRole('ROLE_USER');
            $revokedRole->setPermissions(\openkm\bean\Permission::DELETE);
            $changeSecurity->setRevokedRoles($revokedRoles);

            $this->ws->changeSecurity($changeSecurity);
            echo 'change security';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testChangeSecurity();
?>

```

getProfiles

Description:

Method	Return values	Description
getProfiles(\$filterByActive)	array	Returns the list of all profiles.

Parameters:

\$filterByActive bool type is the parameter filterByActive when enabled the method will return only the active profiles, otherwise will return all available profiles.



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetProfiles() {
        try {
            $profiles = $this->ws->getProfiles(true);
            foreach ($profiles as $profile) {
                var_dump($profile);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetProfiles();
?>
```

getUserProfile

Description:

Method	Return values	Description
getUserProfile(\$userId)	ProfileSimple	Return the profile assigned to a user.

Parameters:

\$user string type is the user

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetUserProfile() {
        try {
            var_dump($this->ws->getUserProfile('okmAdmin'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testGetUserProfile();
?>
```

setUserProfile

Description:

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

Parameters:

\$userId string type is the id user

\$profileId int type is the id profile



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetUserProfile() {
        try {
            $profileId = 1;
            $this->ws->setUserProfile('sochoa', $profileId);
            echo 'set user profile';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleAuth = new ExampleAuth();
$exampleAuth->testSetUserProfile();
?>
```

Conversion samples

Methods

doc2pdf

Description:

Method	Return values	Description
doc2pdf(\$content, \$fileName)	string	Retrieve the uploaded document converted to PDF format.

Parameters:

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.

The openoffice service must be enabled in OpenKM server to get it running.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDoc2pdf($method) {
        try {
            $fileName = dirname(__FILE__) . '/files/test.docx';
            $content = $this->ws->doc2pdf(file_get_contents($fileName), "test.docx");
            switch ($method) {
                case 1:
                    $file = fopen(dirname(__FILE__) . '/files/out.pdf', 'w+');
                    fwrite($file, $content);
            }
        }
    }
}
```

```

        fclose($file);
        echo 'conversion correct';
        break;
    case 2:
        header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
        header('Cache-Control', 'max-age=0, must-revalidate');
        header('Cache-Control', 'post-check=0, pre-check=0');
        header('Pragma', 'no-cache');
        header('Content-Type: application/pdf');
        header('Content-Disposition: attachment; filename="out.pdf"');
        echo $content;
        break;
    }
} catch (Exception $e) {
    var_dump($e);
}
}

}

$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testDoc2pdf(1);
?>

```

pdf2swf

Description:

Method	Return values	Description
pdf2swf(\$content, \$fileName)	string	Retrieve the uploaded document converted to SWF format.

Parameters:

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.



The pdf2swf tool must be enabled in OpenKM server to get it running.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testPdf2swf($method) {
    try {
        $fileName = dirname(__FILE__) . '/files/test.pdf';
        $content = $this->ws->pdf2swf(file_get_contents($fileName), "test.pdf");
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/out.swf', 'w+');
                fwrite($file, $content);
                fclose($file);
                echo 'conversion correct';
                break;
            case 2:
                header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
                header('Cache-Control', 'max-age=0, must-revalidate');
                header('Cache-Control', 'post-check=0, pre-check=0');
                header('Pragma', 'no-cache');
                header('Content-Type: application/x-shockwave-flash');
                header('Content-Disposition: attachment; filename="out.swf"');
                echo $content;
                break;
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testPdf2swf(1)
?>

```

imageConvert

Description:

Method	Return values	Description
imageConvert(\$content, \$fileName, \$params, \$dstMimeType)	string	Retrieve the uploaded image with transformation.
<p>Parameters:</p> <p>\$content string type is recommend using file_get_contents — Reads entire file into a string</p> <p>\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.</p>		

\$params string type

\$dstMimeType string type is the expected document MIME TYPE result.

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

The parameter `dstMimeType` is the expected document MIME TYPE result.



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

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



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

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

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

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testImageConvert() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.png';
            $content = $this->ws->imageConvert(file_get_contents($fileName), "test.png");
            $file = fopen(dirname(__FILE__) . '/files/out.jpg', 'w+');
            fwrite($file, $content);
            fclose($file);
            echo 'image convert';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```

```

}

$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testImageConvert();
?>

```

html2pdf

Description:

Method	Return values	Description
html2pdf(\$url)	string	Retrieve the PDF of an URL.

Parameters:

\$url string type

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

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testHtml2pdf() {
        try {
            $content = $this->ws->html2pdf("http://www.openkm.com");
            $file = fopen(dirname(__FILE__) . '/files/outhtml.pdf', 'w+');
            fwrite($file, $content);
            fclose($file);
            echo 'html to pdf correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```
$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testHtml2pdf();
?>
```

doc2txt

Description:

Method	Return values	Description
doc2txt(\$content, \$fileName)	string	Extracts the text from the upload document.

Parameters:

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDoc2txt() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.docx';
            $content = $this->ws->doc2txt(file_get_contents($fileName), "test.docx");
            echo $content;
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```

```

}

$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testDoc2txt();
?>

```

img2txt

Description:

Method	Return values	Description
img2txt(\$content, \$fileName)	string	Extracts the text from the uploaded image.

Parameters:

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.



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

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testImg2txt() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.png';
            $content = $this->ws->img2txt(file_get_contents($fileName), "test.png");
            echo $content;
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

}

$openkm = new OpenKM(); //autoload
$exampleConversion = new ExampleConversion();
$exampleConversion->testImg2txt();
?>

```

barcode2txt

Description:

Method	Return values	Description
barcode2txt(\$content, \$fileName)	string	Extracts the barcode text from the uploaded image.

Parameters:

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$fileName string type is the document file name. Application uses this parameter to identify by document extension the document MIME TYPE.



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

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleConversion {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO);
    }

    public function testBarcode2txt() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.png';
            $content = $this->ws->barcode2txt(file_get_contents($fileName), "test.png");
            echo $content;
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```
    }  
}  
  
$openkm = new OpenKM(); //autoload  
$exampleConversion = new ExampleConversion();  
$exampleConversion->testBarcode2txt();  
?>
```

Document samples

Basics

On most methods you'll see parameter named "**docId**". The value of this parameter can be a valid document **UUID** or **path**.



Example of docId:

- Using UUID -> "68a27519-c9cc-4490-b281-19ff9f19318b";
- Using path -> "/okm:root/SDK4PHP/logo.png"

Methods

createDocumentSimple

Description:

Method	Return values	Description
createDocumentSimple(\$docPath, \$content)	Document	Creates a new document and returns as a result an object Document with the properties of the created document.
<p>Parameters:</p> <p>\$docPath string type is the Path of the Document</p> <p>\$content string type is recommend using file_get_contents — Reads entire file into a string</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }
}
```

```

public function testCreateDocumentSimple() {
    try {
        $fileName = dirname(__FILE__) . '/files/logo.png';
        $docPath = '/okm:root/SDK4PHP/logo.png';
        $document = $this->ws->createDocumentSimple($docPath, file_get_contents($
        var_dump($document);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCreateDocumentSimple();
?>

```

deleteDocument

Description:

Method	Return values	Description
deleteDocument(\$docId)	void	Deletes a document.

Parameters:

\$docId string type is the uuid or path of the Document

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

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testDeleteDocument(){

```

```

        try {
            $this->ws->deleteDocument('/okm:root/SDK4PHP/logo.png');
            echo 'deleted';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testDeleteDocument();
?>

```

getDocumentProperties

Description:

Method	Return values	Description
getDocumentProperties(\$docId)	Document	Returns the document properties.
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetDocumentProperties() {
        try {
            $document = $this->ws->getDocumentProperties('/okm:root/SDK4PHP/logo.png');
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetDocumentProperties();
?>

```

getContent

Description:

Method	Return values	Description
getContent(\$docId)	string	Retrieves a document content - binary data - of the actual document version.
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetContent($method) {
        $content = $this->ws->getContent('/okm:root/SDK4PHP/logo.png');
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_download.png', 'w+');
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                $document = $this->ws->getDocumentProperties('/okm:root/SDK4PHP/logo.png');
                header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
                header('Cache-Control', 'max-age=0, must-revalidate');
                header('Cache-Control', 'post-check=0, pre-check=0');
                header('Pragma', 'no-cache');
                header('Content-Type: ' . $document->getMimeType());
                header('Content-Disposition: attachment; filename="' . substr($document->getTitle(), 0, 50) . '.png"');
                echo $content;
                break;
        }
    }
}

```

```

    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetContent(2);
?>

```

getContentByVersion

Description:

Method	Return values	Description
getContentByVersion(\$docId, \$versionId)	string	Retrieves a document content (binary data) of some specific document version.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$versionId string type is the uuid or path of the Document</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetContentByVersion($method) {
        $content = $this->ws->getContentByVersion('/okm:root/SDK4PHP/logo.png', 1.1);
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_download_version.png', 'w');
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:

```

```

        $document = $this->ws->getDocumentProperties('/okm:root/SDK4PHP/logo.1
        header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
        header('Cache-Control', 'max-age=0, must-revalidate');
        header('Cache-Control', 'post-check=0, pre-check=0');
        header('Pragma', 'no-cache');
        header('Content-Type: ' . $document->getMimeType());
        header('Content-Disposition: attachment; filename="' . substr($docume:
        echo $content;
        break;
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetContentByVersion(1);
?>

```

getExtractedText

Description:

Method	Return values	Description
getExtractedText(\$docId)	string	Returns a String with the extracted text by text extractor process.

Parameters:

\$docId string type is the uuid or path of the Document

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



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

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



More information at [Statistics](#).

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

```

```

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetExtractedText(){
        try {
            $text = $this->ws->getExtractedText('/okm:root/SDK4PHP/document.odt');
            var_dump($text);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetExtractedText();
?>

```

getThumbnail

Description:

Method	Return values	Description
getThumbnail(\$docId, ThumbnailType \$type)	string	Returns thumbnail image data.

Parameters:

\$docId string type is the uuid or path of the Document

\$type string type

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:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetThumbnail($method) {
        try {
            $content = $this->ws->getThumbnail('/okm:root/SDK4PHP/document.odt', \openkm\util\MimeTypeConfig::MIME_ODT);
            switch ($method) {
                case 1:
                    $file = fopen(dirname(__FILE__) . '/files/thumbnail.png', 'w+');
                    fwrite($file, $content);
                    fclose($file);
                    echo 'download correct';
                    break;
                case 2:
                    header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
                    header('Cache-Control', 'max-age=0, must-revalidate');
                    header('Cache-Control', 'post-check=0, pre-check=0');
                    header('Pragma', 'no-cache');
                    header('Content-Type: ' . \openkm\util\MimeTypeConfig::MIME_PNG);
                    header('Content-Disposition: attachment; filename="thumbnail.png"');
                    echo $content;
                    break;
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetThumbnail(1);
?>

```

getDocumentChildren

Description:

Method	Return values	Description
getDocumentChildren(\$fldId)	array	Returns a list of all documents which their parent is fldId.
Parameters:		

\$docId string type is the uuid or path of the Document or a record node.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetDocumentChildren(){
        try {
            $documents = $this->ws->getDocumentChildren('/okm:root');
            foreach ($documents as $document) {
                var_dump($document);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetDocumentChildren();
?>
```

renameDocument

Description:

Method	Return values	Description
renameDocument(\$docId, \$newName)	Document	Changes the name of a document and returns the Document
Parameters:		
\$docId string type is the uuid or path of the Document		
\$newName string type is the new name for the Document		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRenameDocument() {
        try {
            $document = $this->ws->renameDocument('604e17e8-3285-4e8a-910c-c99ee4e442');
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testRenameDocument();
?>
```

setProperties

Description:

Method	Return values	Description
setProperties(Document \$doc)	void	Change some document properties.
Variables allowed to be changed: <ul style="list-style-type: none"> • Title • Description • Language • Associated categories • Associated keywords 		

The parameter Language must be ISO 691-1 compliant.



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



Only not null and not empty variables will be take on consideration.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetProperties(){
        try {
            $document = $this->ws->getDocumentProperties('604e17e8-3285-4e8a-910c-c990');
            $document->setTitle('Logo');
            $document->setDescription('some description');
            $document->setLanguage('es');
            //Keywords
            $keywords = array();
            $keywords[] = 'test';
            $document->setKeywords($keywords);
            //Categories
            $categories = array();
            $category = $this->ws->getFolderProperties('/okm:categories/test');
            $categories[] = $category;
            $document->setCategories($categories);

            $this->ws->setProperties($document);
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testSetProperties();
?>
```

setLanguage

Description:

Method	Return values	Description
setLanguage(\$docId, \$lang)	void	Sets document language.

Parameters:

\$docId string type is the uuid or path of the Document

\$lang string type is the lang must be ISO 691-1 compliant.

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetLanguage() {
        try {
            $this->ws->setLanguage('/okm:root/SDK4PHP/document.odt', 'es');
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testSetLanguage();
?>
```

setDocumentTitle

Description:

Method	Return values	Description
setDocumentTitle(\$docId, \$title)	void	Set document title.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$title string type is the title of the Document</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetDocumentTitle() {
        try {
            $this->ws->setDocumentTitle('/okm:root/SDK4PHP/document.odt', 'Some title');
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testSetDocumentTitle();
?>
    
```

createFromTemplate

checkout

Description:

Method	Return values	Description

checkout(\$docId)	void	Mark the document for edition.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>Only one user can modify a document at same time.</p> <p>Before starting edition you must perform a checkout action that locks the edition process for other users and allows it only to the user who has executed the action.</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCheckout() {
        try {
            $this->ws->checkout('/okm:root/SDK4PHP/logo.png');
            // At this point the document is locked for other users except for the user who executed the action
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCheckout();
?>
```

cancelCheckout

Description:

Method	Return values	Description

cancelCheckout(\$docId)	void	Cancels a document edition.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p>		
<div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  This action can only be done by the user who previously executed the checkout action. </div>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCancelCheckout() {
        try {
            // At this point the document is locked for other users except for the user who executed the checkout
            $this->ws->cancelCheckout('/okm:root/SDK4PHP/logo.png');
            // At this point other users are allowed to execute a checkout and modify the document
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCancelCheckout();
?>
```

forceCancelCheckout

Description:

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

Parameters:

\$docId string type is the uuid or path of the Document

This method allows to cancel edition on any document.

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


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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testForceCancelCheckout() {
        try {
            // At this point the document is locked for other users except for the user
            $this->ws->forceCancelCheckout('/okm:root/SDK4PHP/logo.png');
            // At this point other users are allowed to execute a checkout and modify
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testForceCancelCheckout();
?>
    
```

isCheckedOut

Description:

Method	Return values	Description
isCheckedOut(\$docId)	bool	Returns a boolean that indicate if the document is on edition or not.

<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p>

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testIsCheckedOut() {
        try {
            echo "Is the document checkout:" . $this->ws->isCheckedOut('/okm:root/SDK');
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testIsCheckedOut();
?>
```

checkin

Description:

Method	Return values	Description
checkin(\$docId, \$content, \$comment,\$increment)	Version	Updates a document with new version and return an object with new Version values.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p>		

\$content string type is recommend using `file_get_contents` — Reads entire file into a string

\$comment string type is the comment for the new version the document

\$increment int type



The value of increment variable must be 1 or greater.

The valid values of increment variable depends on the `VersionNumberAdapter` you have enabled.



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCheckin() {
        try {
            $fileName = dirname(__FILE__) . '/files/logo.png';
            $version = $this->ws->checkin('/okm:root/SDK4PHP/logo.png', file_get_contents($fileName));
            var_dump($version);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCheckin();
?>
```

getVersionHistory

Description:

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

getVersionHistory(\$docId)	array	Returns a list of all document versions.
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

//ini_set('display_errors', true);
//error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetVersionHistory() {
        try {
            $versions = $this->ws->getVersionHistory('/okm:root/SDK4PHP/logo.png');
            foreach ($versions as $version) {
                var_dump($version);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetVersionHistory();
?>
```

lock

Description:

Method	Return values	Description
lock(\$docId)	LockInfo	Locks a document and returns an object with the Lock information.

Parameters:

\$docId string type is the uuid or path of the Document



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

A locked document can not be modified by other users.

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testLock() {
        try {
            $lockInfo = $this->ws->lock('/okm:root/SDK4PHP/logo.png');
            var_dump($lockInfo);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testLock();
?>
```

unlock

Description:

Method	Return values	Description
unlock(\$docId)	void	Unlocks a locked document.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p>		


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

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUnlock() {
        try {
            $this->ws->unlock('/okm:root/SDK4PHP/logo.png');
            echo 'unlock';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testUnlock();
?>
    
```

forceUnlock

Description:

Method	Return values	Description
forceUnlock(\$docId)	void	Unlocks a locked document.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>This method allows to unlcok any locked document.</p> <p>It is not mandatory execute this action by the same user who previously executed the checkout lock action.</p>		



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testForceUnlock() {
        try {
            $this->ws->forceUnlock('/okm:root/SDK4PHP/logo.png');
            echo 'forceUnlock';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testForceUnlock();
?>
```

isLocked

Description:

Method	Return values	Description
isLocked(\$docId)	bool	Returns a boolean that indicate if the document is locked or not.
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testIsLocked() {
        try {
            echo "Is document locked:" . $this->ws->isLocked('/okm:root/SDK4PHP/logol.}
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testIsLocked();
?>

```

getLockInfo

Description:

Method	Return values	Description
getLockInfo(\$docId)	LockInfo	Returns an object with the Lock information
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetLockInfo() {
    try {
        var_dump($this->ws->getLockInfo('/okm:root/SDK4PHP/logo.png'));
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetLockInfo();
?>

```

purgeDocument

Description:

Method	Return values	Description
purgeDocument(\$docId)	void	Document is definitely removed from repository.

Parameters:

\$docId string type is the uuid or path of the Document

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

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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testPurgeDocument() {
    try {
        $this->ws->purgeDocument('/okm:root/SDK4PHP/logo.png');
        echo 'purgeDocument';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testPurgeDocument();
?>

```

moveDocument

Description:

Method	Return values	Description
moveDocument(\$docId, \$dstId)	void	Move a document into some folder or record.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$dstId string type is the uuid or path of the Folder or Record</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

```

```

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testMoveDocument() {
    try {
        $this->ws->moveDocument('/okm:root/SDK4PHP/logo.png', '/okm:root/SDK4PHP/test');
        echo 'moveDocument';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testMoveDocument();
?>

```

copyDocument

Description:

Method	Return values	Description
copyDocument(\$docId, \$dstId, \$newName)	void	Copies a document to a folder or record.

Parameters:

\$docId string type is the uuid or path of the Document

\$dstId string type is the uuid or path of the Folder or Record

\$newName string type is the new name for the Document

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



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

See "**extendedDocumentCopy**" method for this feature.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
}

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO:
}

public function testCopyDocument() {
    try {
        $this->ws->copyDocument('/okm:root/SDK4PHP/logo.png', '/okm:root/SDK4PHP/T1
        echo 'copyDocument';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCopyDocument();
?>

```

restoreVersion

Description:

Method	Return values	Description
restoreVersion(\$docId, \$versionId)	void	Promotes previous document's version to the actual version.

Parameters:

\$docId string type is the uuid or path of the Document

\$versionId string type is the version of the Document

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO:

```

```

    }

    public function testRestoreVersion() {
        try {
            $this->ws->restoreVersion('/okm:root/SDK4PHP/logo.png', '1.1');
            echo 'restoreVersion';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testRestoreVersion();
?>

```

purgeVersionHistory

Description:

Method	Return values	Description
purgeVersionHistory(\$docId)	void	Purges all documents version except the actual version.

Parameters:

\$docId string type is the uuid or path of the Document

This action compact the version history of a document.

 This action can not be reverted.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testPurgeVersionHistory() {

```

```

        try {
            // Version history has version 1.3,1.2,1.1 and 1.0
            $this->ws->purgeVersionHistory('/okm:root/SDK4PHP/logo.png');
            // Version history has only version 1.3
            echo 'purgeVersionHistory';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testPurgeVersionHistory();
?>

```

getVersionHistorySize

Description:

Method	Return values	Description
getVersionHistorySize(\$docId)	int	Returns the sum in bytes of all documents into documents history.

Parameters:

\$docId string type is the uuid or path of the Document

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetVersionHistorySize() {
        try {
            $units = array("B", "KB", "MB", "GB", "TB", "PB", "EB");

            $bytes = $this->ws->getVersionHistorySize('/okm:root/SDK4PHP/logo.png');
            $value = "";

            for ($i = 6; $i > 0; $i--) {

```

```

        $step = pow(1024, $i);
        if ($bytes > $step) {
            $value = number_format($bytes / $step, 2) . ' ' . $units[$i];
        }
        if (empty($value)) {
            $value = $bytes . ' ' . $units[0];
        }
    }
    echo $value;
} catch (Exception $e) {
    var_dump($e);
}
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetVersionHistorySize();
?>

```

isValidDocument

Description:

Method	Return values	Description
isValidDocument(\$docId)	bool	Returns a boolean that indicates if the node is a document or not.
Parameters:		
\$docId string type is the uuid or path of the Document		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testIsValidDocument() {
        try {
            // Return true
            var_dump($this->ws->isValidDocument("/okm:root/SDK4PHP/logo.png"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

        // Return false
        var_dump($this->ws->isValidDocument('/okm:root'));
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testIsValidDocument();
?>

```

getDocumentPath

Description:

Method	Return values	Description
getDocumentPath(\$uuid)	string	Converts a document UUID to a document path.
Parameters:		
\$uuid string type is the uuid of the Document		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetDocumentPath() {
        try {
            var_dump($this->ws->getDocumentPath("8b559709-3c90-4c26-b181-5192a17362b2"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload

```

```
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetDocumentPath();
?>
```

getDetectedLanguages

Description:

Method	Return values	Description
getDetectedLanguages()	array	Return a list of available document languages what OpenKM can identify

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetDetectedLanguages() {
        try {
            var_dump($this->ws->getDetectedLanguages());
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetDetectedLanguages();
?>
```

extendedDocumentCopy

Description:

Method	Return values	Description
extendedDocumentCopy(\$docId, \$dstId, \$name, \$categories,	void	Copies a document with

<p>\$keywords, \$propertyGroups, \$notes, \$wiki)</p>	<p>associated data into a folder or record.</p>
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$dstId string type is the uuid or path of the Folder or Record</p> <p>\$name string type is the new name for the Document. Value is null, document will preserve the same name.</p> <p>\$categories bool type</p> <p>\$keywords bool type</p> <p>\$propertyGroups bool type</p> <p>\$notes bool type</p> <p>\$wiki bool type</p> <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <p>i By default only the binary data and the security grants, the metadata, keywords, etc. of the document are not copied.</p> <p>Additional:</p> <ul style="list-style-type: none"> • When the category parameter is true the original values of the categories will be copied. • When the keywords parameter is true the original values of the keywords will be copied. • When the propertyGroups parameter is true the original values of the metadata groups will be copied. • When the notes parameter is true the original values of the notes will be copied. • When the wiki parameter is true the original values of the wiki will be copied. </div>	

Example:

```
<?php
```

```

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testExtendedDocumentCopy() {
        try {
            $this->ws->extendedDocumentCopy('/okm:root/SDK4PHP/logo.png', '/okm:root/');
            echo 'extendedDocumentCopy';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testExtendedDocumentCopy();
?>

```

createFromTemplate

Description:

Method	Return values	Description
<p>Document createFromTemplate(\$docId, \$dstPath,\$categories,\$keywords, \$propertyGroups, \$notes, \$wiki, \$formElements)</p>	<p>Document</p>	<p>Creates a new document from template and returns an object Document.</p>
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$dstPath string type is the path of the Document</p> <p>\$categories bool type</p> <p>\$keywords bool type</p> <p>\$propertyGroups bool type</p>		

\$notes bool type

\$wiki bool type

\$formElements array type is an array of the FormElement

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



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



Additional:

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

Example:



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

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCreateFromTemplate() {
        try {
            $formElements = array();
        }
    }
}
```

```

$name = new \openkm\bean\form\Input();
$name->setName('okp:tpl.name');
$name->setValue('Some name');
$formElements[] = $name;

$birdDate = new \openkm\bean\form\Input();
$birdDate->setName('okp:tpl.bird_date');
// Value must be converted to String ISO 8601 compliant
$birdDate->setValue(date('Ymdhis'));
$formElements[] = $birdDate;

$language = new \openkm\bean\form>Select();
$language->setName('okp:tpl.language');

$options = array();
$options[] = new \openkm\bean\form\Option();
$options[0]->setValue('php');
$options[0]->setSelected(true);
$language->setOptions($options);
$formElements[] = $language;

try {
    $document = $this->ws->createFromTemplate('/okm:templates/tpl.pdf', '/okm
    var_dump($document);
} catch (Exception $e) {
    var_dump($e);
}
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCreateFromTemplate();
?>

```

createFromTemplateSimple

Description:

Method	Return values	Description
<p>Document <code>createFromTemplateSimple(\$docId, \$dstPath, \$categories, \$keywords, \$propertyGroups, \$notes, \$wiki, \$properties)</code></p>	<p>Document</p>	<p>Creates a new document from template and returns an object Document.</p>
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$dstPath string type is the path of the Document</p> <p>\$categories bool type</p>		

\$keywords bool type

\$propertyGroups bool type

\$notes bool type

\$wiki bool type

\$properties array type is an array of the SimplePropertyGroup

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



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



Additional:

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

Example:



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

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}
```

```

public function testCreateFromTemplateSimple() {
    try {
        $properties = array();

        $name = new \openkm\bean\SimplePropertyGroup();
        $name->setName('okp:tpl.name');
        $name->setValue('Some name');
        $properties[] = $name;

        $birdDate = new \openkm\bean\SimplePropertyGroup();
        $birdDate->setName('okp:tpl.bird_date');
        // Value must be converted to String ISO 8601 compliant
        $birdDate->setValue(date('Ymdhis'));
        $properties[] = $birdDate;

        $language = new \openkm\bean\SimplePropertyGroup();
        $language->setName('okp:tpl.language');
        $language->setValue('php');
        $properties[] = $language;

        $document = $this->ws->createFromTemplateSimple('/okm:templates/tpl.pdf',
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testCreateFromTemplateSimple();
?>

```

updateFromTemplate

Description:

Method	Return values	Description
Document updateFromTemplate(\$docId, \$dstId, \$formElements)	Document	Update a document previously created from template and returns an object Document.

Parameters:

\$docId string type is the uuid or path of the Document

\$dstId string type is the uuid or path of the Document

\$formElements array type is an array of the FormElement

This method only has sense when template use metadata groups to fill fields into.


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

Example:



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

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUpdateFromTemplate() {
        try {
            $formElements = array();

            $name = new \openkm\bean\form\Input();
            $name->setName('okp:tpl.name');
            $name->setValue('Update name');
            $formElements[] = $name;

            $birdDate = new \openkm\bean\form\Input();
            $birdDate->setName('okp:tpl.bird_date');
            // Value must be converted to String ISO 8601 compliant
            $birdDate->setValue(date('Ymdhis'));
            $formElements[] = $birdDate;

            $language = new \openkm\bean\form>Select();
            $language->setName('okp:tpl.language');

            $options = array();
            $option = new \openkm\bean\form\Option();
            $option->setValue('go');
            $option->setSelected(true);
            $options[] = $option;
            $language->setOptions($options);
            $formElements[] = $language;

            $document = $this->ws->updateFromTemplate('/okm:templates/tpl.pdf', '/okm:templates/tpl.pdf');
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testUpdateFromTemplate();
```

?>

updateFromTemplateSimple

Description:

Method	Return values	Description
updateFromTemplateSimple(String docId, String dstId, Map<String, String> properties)	Document	Update a document previously created from template and return an object Document.

Parameters:

\$docId string type is the uuid or path of the Document

\$dstId string type is the uuid or path of the Document

\$properties array type is an array of the SimplePropertyGroup

This method only has sense when template use metadata groups to fill fields into.

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

Example:



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

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUpdateFromTemplateSimple() {
        try {
            $properties = array();
        }
    }
}
```

```

        $name = new \openkm\bean\SimplePropertyGroup();
        $name->setName('okp:tpl.name');
        $name->setValue('Update name');
        $properties[] = $name;

        $birdDate = new \openkm\bean\SimplePropertyGroup();
        $birdDate->setName('okp:tpl.bird_date');
        // Value must be converted to String ISO 8601 compliant
        $birdDate->setValue(date('Ymdhis'));
        $properties[] = $birdDate;

        $language = new \openkm\bean\SimplePropertyGroup();
        $language->setName('okp:tpl.language');
        $language->setValue('go');
        $properties[] = $language;

        $document = $this->ws->updateFromTemplateSimple('/okm:templates/tpl.pdf',
            var_dump($document);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testUpdateFromTemplateSimple();
?>

```

getAnnotations

Description:

Method	Return values	Description
getAnnotations(\$docId, \$verName)	string	Return the document annotations of some document version.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$verName string type is the version name of the Document</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetAnnotations(){
    try {
        var_dump($this->ws->getAnnotations('e34af683-7a8f-447c-98fb-8417b399f8d5'));
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetAnnotations();
?>

```

getDifferences

Description:

Method	Return values	Description
getDifferences(\$docId, \$v1, \$v2)	string	Return a PDF document with the differences between two document versions.
<p>Parameters:</p> <p>\$docId string type is the uuid or path of the Document</p> <p>\$v1 string type is the version of the Document</p> <p>\$v2 string type is the version of the Document</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
}

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetDifferences($method) {
    try {
        $content = $this->ws->getDifferences('e34af683-7a8f-447c-98fb-8417b399f8d');
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/text.csv', 'w+');
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                $document = $this->ws->getDocumentProperties('/test.csv');
                header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
                header('Cache-Control', 'max-age=0, must-revalidate');
                header('Cache-Control', 'post-check=0, pre-check=0');
                header('Pragma', 'no-cache');
                header('Content-Type: ' . $document->getMimeType());
                header('Content-Disposition: attachment; filename="' . substr($document->getTitle(), 0, 50) . '.csv"');
                echo $content;
                break;
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetDifferences(1);
?>

```

getCheckedOut

Description:

Method	Return values	Description
getCheckedOut()	array	Return a list of documents checkout by the user.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

    const HOST = "http://localhost:8080/OpenKM/";
}

```

```
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetCheckedOut() {
    try {
        $documents = $this->ws->getCheckedOut();
        foreach ($documents as $document) {
            var_dump($document);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleDocument = new ExampleDocument();
$exampleDocument->testGetCheckedOut();
?>
```

Folder samples

Basics

On almost methods you'll see parameter named "**fldId**". The value of this parameter can be some valid folder **UUID** or **path**.



Example of fldId:

- Using UUID -> "6e86cc0b-bc9a-4c51-8296-e9d4e749e449";
- Using path -> "/okm:root/SDK4PHP/test"

Methods

createFolder

Description:

Method	Return values	Description
createFolder(Folder \$fld)	Folder	Creates a new folder and returns a result an object Folder.

The variable **path** into the parameter **\$fld**, must be initialized. It indicates the folder path into OpenKM.

```
Folder fld = new Folder();
fld.setPath("/okm:root/SDK4PHP/test");
```

 The other variables of a Folder (fld) will not take any effect on folder creation.

We suggest using the method below to create a FolderSimple rather this one.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testCreateFolder() {
    try {
        $fld = new Folder();
        $fld->setPath("/okm:root/SDK4PHP/test");
        $folder = $this->ws->createFolder($fld);
        var_dump($folder);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testCreateFolder();
?>

```

createFolderSimple

Description:

Method	Return values	Description
createFolderSimple(\$fldPath)	Folder	Creates a new folder and return it as a result of an object Folder.

Parameters:
\$fldPath string type is the Path of the Folder

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}

```

```

    public function testCreateFolderSimple() {
        try {
            $folder = $this->ws->createFolderSimple("/okm:root/SDK4PHP/test");
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testCreateFolderSimple();

?>

```

getFolderProperties

Description:

Method	Return values	Description
getFolderProperties(\$fldId)	Folder	Returns the folder properties.
Parameters:		
\$fldId string type is the uuid or path of the Folder		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO:
    }

    public function testGetFolderProperties() {
        try {
            $folder = $this->ws->getFolderProperties("/okm:root/SDK4PHP/test");
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testGetFolderProperties();

?>

```

deleteFolder

Description:

Method	Return values	Description
deleteFolder(\$fldId)	void	Deletes a folder.
<p>Parameters:</p> <p>\$fldId string type is the uuid or path of the Folder</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteFolder() {
        try {
            $this->ws->deleteFolder("/okm:root/SDK4PHP/test");
            echo 'delete folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testDeleteFolder();

```

```
?>
```

renameFolder

Description:

Method	Return values	Description
renameFolder(\$fldId, \$newName)	void	Renames a folder.
<p>Parameters:</p> <p>\$fldId string type is the uuid or path of the Folder</p> <p>\$newName string type is the new name for the Folder</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRenameFolder() {
        try {
            // Exists folder /okm:root/SDK4PHP/test
            $this->ws->renameFolder("/okm:root/SDK4PHP/test", "renamedFolder");
            // Folder has renamed to /okm:root/SDK4PHP/renamedFolder
            echo 'rename Folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testRenameFolder();
?>
```

moveFolder

Description:

Method	Return values	Description
moveFolder(\$fldId, \$dstId)	void	Moves a folder into other folder or record.
Parameters:		
\$fldId string type is the uuid or path of the Folder		
\$dstId string type is the uuid or path of the Folder or record		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testMoveFolder(){
        try {
            // Exists folder /okm:root/SDK4PHP/test
            $this->ws->moveFolder("/okm:root/SDK4PHP/test", "/okm:root/SDK4PHP/tmp");
            // Folder has moved to /okm:root/SDK4PHP/tmp/test
            echo 'move Folder';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testMoveFolder();

?>

```

getFolderChildren

Description:

Method	Return values	Description
getFolderChildren(\$fldId)	array	Returns an array of all folders which their parent is fldId.
<p>Parameters:</p> <p>\$fldId string type is the uuid or path of the Folder or Record node</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetFolderChildren(){
        try {
            $folders = $this->ws->getFolderChildren("/okm:root/SDK4PHP");
            foreach ($folders as $folder) {
                var_dump($folder);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testGetFolderChildren();

?>
```

isValidFolder

Description:

Method	Return values	Description

isValidFolder(\$fldId)	Boolean	Returns a boolean that indicates if the node is a folder or not.
Parameters:		
\$fldId string type is the uuid or path of the Folder		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testIsValidFolder(){
        try {
            // Return false
            var_dump($this->ws->isValidFolder("/okm:root/SDK4PHP/logo.png"));
            // Return true
            var_dump($this->ws->isValidFolder("/okm:root/SDK4PHP"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testIsValidFolder();

?>
```

getFolderPath

Description:

Method	Return values	Description
getFolderPath(\$uuid)	String	Convert folder UUID to folder path.
Parameters:		

\$uuid string type is the uuid of the Folder

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetFolderPath(){
        try {
            var_dump($this->ws->getFolderPath("6e86cc0b-bc9a-4c51-8296-e9d4e749e449"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testGetFolderPath();

?>
```

copyFolder

Description:

Method	Return values	Description
copyFolder(\$fldId, \$dstId, \$newName)	void	Copies a folder into other folder or record.
Parameters:		
\$fldId string type is the uuid or path of the Folder		
\$dstId string type is the uuid or path of the Folder or Record		
\$newName string type is the new name for the Folder. Value is null, folder will preservates the same name.		



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

See "**extendedFolderCopy**" method for this feature.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCopyFolder(){
        try {
            $this->ws->copyFolder("/okm:root/SDK4PHP/test", "/okm:root/SDK4PHP/tmp", "n");
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testCopyFolder();

?>
```

extendedFolderCopy

Description:

Method	Return values	Description
extendedFolderCopy(\$fldId, \$dstId, \$categories, \$keywords, \$propertyGroups, \$notes, \$wiki)	void	Copies a folder with associated data into other folder or record.
Parameters:		
\$fldId string type is the uuid or path of the Folder		

\$dstId string type is the uuid or path of the Folder or Record

\$categories boolean type

\$keywords boolean type

\$propertyGroups boolean type

\$notes boolean type

\$wiki boolean type



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

Additionally:

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testExtentedFolderCopy() {
        try {
            $this->ws->extendedFolderCopy("/okm:root/SDK4PHP/test", "/okm:root/SDK4PHP/test");
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```

```

}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testExtentedFolderCopy();
?>
    
```

getContentInfo

Description:

Method	Return values	Description
getContentInfo(\$fldId)	ContentInfo	Return and object ContentInfo with information about folder.

Parameters:

\$fldId string type is the uuid or path of the Folder.

The ContentInfo object retrieves information about:

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

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetContentInfo(){
    
```

```

        try {
            var_dump($this->ws->getContentInfo("/okm:root/SDK4PHP"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testGetContentInfo();
?>

```

purgeFolder

Description:

Method	Return values	Description
purgeFolder(\$fldId)	void	Folder is definitely removed from repository.

Parameters:
\$fldId string type is the uuid or path of the Folder.

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



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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }
}

```

```

    public function testPurgeFolder() {
        try {
            $this->ws->purgeFolder("/okm:root/SDK4PHP/test");
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testPurgeFolder();
?>

```

setStyle

Description:

Method	Return values	Description
setStyle(\$fIdId, \$styleId)	void	Set the folder style.

Parameters:

\$fIdId string type is the uuid or path of the Folder.

\$styleId long type is the id of the style.

 More information at: [Folder style](#).

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testSetStyle() {
        try {

```

```

        $this->ws->setStyle("/okm:root/SDK4PHP/test",1);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testSetStyle();
?>

```

createMissingFolders

Description:

Method	Return values	Description
createMissingFolders(\$fldPath)	void	Create missing folders.
Parameters:		
\$fldPath string type is the path of the Folder.		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebservices;
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Folder;

class ExampleFolder {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO:
    }

    public function testCreateMissingFolders() {
        try {
            $this->ws->createMissingFolders("/okm:root/missingfld1/missingfld2/missin:
            var_dump('create missing folders');
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}
}

```

```
$openkm = new OpenKM(); //autoload
$exampleFolder = new ExampleFolder();
$exampleFolder->testCreateMissingFolders()
?>
```

Mail samples

Basics

On most methods you'll see parameter named "**mailId**". The value of this parameter can be a valid mail **UUID** or **path**.

 Example of fldId:

- Using UUID -> "520f4d98-1855-4784-9bc3-01ba5b440c1d";
- Using path -> "/okm:root/2937b81d-0b10-4dd0-a426-9acbd80be1c9-some subject"

Methods

createMail

Description:

Method	Return values	Description
createMail(Mail \$mail)	Mail	Creates a new mail and returns as a result an object Mail.

The variable **path** into the parameter **mail**, must be initialized. It indicates the folder path into OpenKM.

Other mandatory variables:

- size (mail size in bytes).
- from (mail from account).
- reply, to, cc, bcc (mail accounts are optional).
- sendDate (date when mail was sent).
- receivedDate (date when was received).
- subject (mail subject).
- content (the mail content).
- mimeType (HTML or text mime type).
- headers (the mail headers are optional).
- raw (the mail raw are optional).
- origin (msg, eml, api, pop3, imap origin)
- title (mail title)

 Mails account allowed formats:

- "\"John King\" <jking@mail.com>"
- "<jking@mail.com>"



Mail path allowed is:

MSGID . "-" . sanitized(subject).



MIME types values:

- Mail::MIME_TEXT for text mail format.
- Mail::MIME_HTML for html mail format.

Origin values:

- Mail::ORIGIN_MSG for .msg mail format.
- Mail::ORIGIN_EML for .eml mail format.
- Mail::ORIGIN_API for mail format created from API.
- Mail::ORIGIN_POP3 for mail imported from pop3 account.
- Mail::ORIGIN_IMAP for mail imported from imap account.



The other variables of Mail (mail) will not take any effect on mail creation.

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCreateMail() {
        try {
            $mail = new \openkm\bean\Mail();
            // Mail path = msgId + escaped(subject)
            $msgId = "2937b81d-0b10-4dd0-a426-9acbd80be1c9";
            $subject = "some subject";
        }
    }
}
```

```

        $mailPath = "/okm:root/SDK4PHP/" . $msgId . "-" . $this->escape($subject);
        $mail->setPath($mailPath);
        // Other format for mail "some name <no_reply@openkm.com>"
        $mail->setFrom('<no_reply@openkm.com>');
        $mail->setTo(['anonymous@gmail.com', 'anonymous1@gmail.com']);
        // You should set real dates
        $mail->setSentDate(date('Y-m-d'));
        $mail->setReceivedDate(date('Y-m-d'));
        $mail->setContent('some content');
        $mail->setMimeType(\openkm\bean\Mail::MIME_TEXT);
        $mail->setSubject($subject);
        $mail->setOrigin(\openkm\bean\Mail::ORIGIN_API);
        // Get only as an approximation of real size for these sample
        $mail->setSize(strlen($mail->toString()));
        var_dump($this->ws->createMail($mail));
    } catch (Exception $e) {
        var_dump($e);
    }
}

private function escape($name) {
    $ret = $this->cleanup($name);
    return htmlentities($ret);
}

private function cleanup($name) {
    $ret = str_replace('/', ' ', $name);
    $ret = str_replace('*', ' ', $ret);
    $ret = str_replace('\s+', ' ', $ret);
    return $ret;
}

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testCreateMail();
?>

```

getMailProperties

Description:

Method	Return values	Description
getMailProperties(\$mailId)	Mail	Returns the mail properties.
Parameters:		
\$mailId string type is the uuid or path of the Mail		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

```

```

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetMailProperties() {
        try {
            $mail = $this->ws->getMailProperties('520f4d98-1855-4784-9bc3-01ba5b440c1');
            var_dump($mail);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testGetMailProperties();
?>

```

deleteMail

Description:

Method	Return values	Description
deleteMail(\$mailId)	void	Deletes a mail.
Parameters:		
\$mailId string type is the uuid or path of the Mail		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testDeleteMail() {
    try {
        $this->ws->deleteMail('520f4d98-1855-4784-9bc3-01ba5b440c1d');
        echo 'deleted';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testDeleteMail();
?>

```

renameMail

Description:

Method	Return values	Description
renameMail(\$mailId, \$newName)	Mail	Change the name of a Mail and returns the Mail

Parameters:

\$mailId string type is the uuid or path of the Mail

\$newName string type is the new name for the Mail

i From OpenKM frontend UI the subject is used to show the mail name at file browser table. That means this change will take effect internally on mail path, but will not be appreciated from default UI.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

```

```

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
}

public function testRenameMail() {
    try {
        $mail = $this->ws->renameMail('520f4d98-1855-4784-9bc3-01ba5b440c1d', 'ne
        var_dump($mail);
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testRenameMail();
?>

```

moveMail

Description:

Method	Return values	Description
moveMail(\$mailId, \$dstId)	void	Moves mail into a folder or record.
<p>Parameters:</p> <p>\$mailId string type is the uuid or path of the Mail</p> <p>\$dstId string type is the uuid or path of the Folder or Record</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testMoveMail() {
        try {

```

```

        $this->ws->moveMail('520f4d98-1855-4784-9bc3-01ba5b440c1d', '/okm:root/SD
        echo 'move';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testMoveMail();
?>

```

getMailChildren

Description:

Method	Return values	Description
getMailChildren(\$fldId)	array	Returns a list of all mails which their parent is fldId.
Parameters:		
\$fldId string type is the uuid or path of the Folder or a record node.		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testGetMailChildren() {
        try {
            $mails = $this->ws->getMailChildren('/okm:root/SDK4PHP');
            foreach ($mails as $mail) {
                var_dump($mail);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testGetMailChildren();
?>

```

isValidMail

Description:

Method	Return values	Description
isValidMail(\$mailId)	bool	Returns a boolean that indicate if the node is a mail or not.
Parameters:		
\$mailId string type is the uuid or path of the Mail		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testIsValidMail() {
        try {
            //Return false
            var_dump($this->ws->isValidMail('/okm:root/SDK4PHP/logo.png'));
            //Return true
            var_dump($this->ws->isValidMail('520f4d98-1855-4784-9bc3-01ba5b440c1d'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testIsValidMail();
?>

```

getMailPath

Description:

Method	Return values	Description
getMailPath(\$uuid)	string	Converts mail UUID to mail path.
Parameters:		
\$uuid string type is the uuid of the Mail		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetMailPath() {
        try {
            var_dump($this->ws->getMailPath('520f4d98-1855-4784-9bc3-01ba5b440c1d'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testGetMailPath();
?>
```

createAttachment

Description:

Method	Return values	Description

createAttachment(\$mailId, \$docName, \$content)	Document	Stores into the mail an attachment and returns an object Document with the properties of the created attachment.
Parameters:		
\$mailId string type is the uuid or path of the Mail		
\$docName string type is the Name of the Attachment		
\$content string type is recommend using file_get_contents — Reads entire file into a string		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCreateAttachment() {
        try {
            $fileName = dirname(__FILE__) . '/files/logo.png';
            $docName = 'logo.png';
            $document = $this->ws->createAttachment('520f4d98-1855-4784-9bc3-01ba5b44');
            var_dump($document);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testCreateAttachment();
?>
```

deleteAttachment

Description:

Method	Return values	Description

deleteAttachment(\$mailId, \$docId)	void	Deletes a mail attachment.
<p>Parameters:</p> <p>\$mailId string type is the uuid or path of the Mail</p> <p>\$docId string type is the uuid or path of the Document</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteAttachment() {
        try {
            $this->ws->deleteAttachment('520f4d98-1855-4784-9bc3-01ba5b440c1d', '73ba
            echo 'deleted attachment';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testDeleteAttachment();
?>
```

getAttachments

Description:

Method	Return values	Description
getAttachments(String mailId)	array	Retrieves a list of all documents attachment of a mail.
<p>Parameters:</p>		

\$mailId string type is the uuid or path of the Mail

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetAttachments() {
        try {
            $documents = $this->ws->getAttachments('520f4d98-1855-4784-9bc3-01ba5b440...');
            foreach ($documents as $document) {
                var_dump($document);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testGetAttachments();
?>
```

purgeMail

Description:

Method	Return values	Description
purgeMail(\$mailId)	void	Mail is definitely removed from repository.

Parameters:

\$mailId string type is the uuid or path of the Mail

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





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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testPurgeMail() {
        try {
            $this->ws->purgeMail('520f4d98-1855-4784-9bc3-01ba5b440c1d');
            echo 'purge';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testPurgeMail();
?>
```

copyMail

Description:

Method	Return values	Description
public void copyMail(\$mailId, \$dstId, \$newName)	void	Copies mail into a folder or record.

Parameters:

\$mailId string type is the uuid or path of the Mail

\$dstId string type is the uuid or path of the Folder or Record

\$newName string type is the new name for the Document

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



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

See "**extendedMailCopy**" method for this feature.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCopyMail() {
        try {
            $this->ws->copyMail('520f4d98-1855-4784-9bc3-01ba5b440c1d', '/okm:root/SD');
            echo 'copyMail';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testCopyMail();
?>
```

extendedMailCopy

Description:

Method	Return values	Description
extendedMailCopy(\$mailId, \$dstId, \$categories, \$keywords, \$propertyGroups, \$notes, \$wiki, \$newName)	void	Copies mail width with associated data into a folder or record.

Parameters:

\$mailId string type is the uuid or path of the Mail

\$dstId string type is the uuid or path of the Folder or Record

\$categories bool type

\$keywords bool type

\$propertyGroups bool type

\$notes bool type

\$wiki bool type

\$newName string type is the new name for the Mail. Value is null, mail will preserve the same name.



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

Additional:

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}
```

```

public function testExtendedMailCopy() {
    try {
        $this->ws->extendedMailCopy('3c19ceb2-9c6a-4b43-aabb-7f169127022a', '/okm
        echo 'extendedMailCopy';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testExtendedMailCopy();
?>

```

sendMailWithAttachments

Description:

Method	Return values	Description
sendMailWithAttachments(\$from, \$subject, \$body, \$dstId, \$toRecipients, \$ccRecipients, \$bccRecipients, \$docsId)	void	Sends mail message with attachment.
<p>Parameters:</p> <p>\$from string the mail from</p> <p>\$subject string type is the mail subject</p> <p>\$body string type is the message of the email</p> <p>\$dstId string type is the uuid or path of the Folder or Record</p> <p>\$toRecipients array type are a array of mail accounts destination</p> <p>\$ccRecipients array type are a array of mail accounts destination (optinal)</p> <p>\$bccRecipients array type are a array of mail accounts destination (optional)</p> <p>\$docsId array type are a array of valid document UUID already into OpenKM that will be send as attachment into mail (optional).</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

```

```

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSendMailWithAttachments() {
        try {
            $to = ['destination@gmail.com', 'destination1@gmail.com'];
            $cc = ['destination3@gmail.com', 'destination4@gmail.com'];
            $bcc = ['destination5@gmail.com', 'destination6@gmail.com'];
            $docsId = ['f123a950-0329-4d62-8328-0ff500fd42db', '/okm:root/SDK4PHP/logo'];
            $this->ws->sendMailWithAttachments('sochoa@openkm.com', 'some subject', 'some content');
            echo 'sendMailWithAttachments';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testSendMailWithAttachments();
?>

```

importEml

Description:

Method	Return values	Description
importEml(\$dstId, \$title, \$content)	Mail	Import a mail in EML format.

Parameters:

\$dstId string type is the uuid or path of the Folder or Record. The dstId parameter indicate where the mail will be stored in the repository after is sent.

\$title string type is the title of the Mail.

\$content string type is recommend using file_get_contents — Reads entire file into a string

Example:

```
<?php
```

```

include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testImportEml() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.eml';
            $mail = $this->ws->importEml('768748c5-04ab-4921-b0d4-7545d239ce5a', 'some');
            var_dump($mail);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testImportEml();
?>

```

importMsg

Description:

Method	Return values	Description
importMsg(\$dstId, \$title, \$content)	Mail	Import a mail in MSG format.
<p>Parameters:</p> <p>\$dstId string type is the uuid or path of the Folder or Record. The dstId parameter indicate where the mail will be stored in the repository after is sent.</p> <p>\$title string type is the title of the Mail.</p> <p>\$content string type is recommend using file_get_contents — Reads entire file into a string</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testImportMsg() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.eml';
            $mail = $this->ws->importEml('702e456a-3145-4dd5-831d-01647a9f2608', 'some');
            var_dump($mail);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testImportMsg();
?>

```

setMailTitle

Description:

Method	Return values	Description
setMailTitle(\$mailId, \$title)	void	Import a mail in MSG format.
<p>Parameters:</p> <p>\$mailId string type is the uuid or path of the Mail</p> <p>\$title string type is the title of the Mail.</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

```

```

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetMailTitle() {
        try {
            $this->ws->setMailTitle('3c19ceb2-9c6a-4b43-aabb-7f169127022a', 'some title');
            echo 'mail title';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testSetMailTitle();
?>

```

sendMail

Description:

Method	Return values	Description
sendMail(\$subject, \$body, \$recipients)	void	Sends a mail.
<p>Parameters:</p> <p>\$subject string type is the mail subject</p> <p>\$body string type is the message of the email.</p> <p>\$recipients string type is array of mail accounts destination</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleMail {

```

```
const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testSendMail() {
    try {
        $recipients = array();
        $recipients[] = 'gnu.java.sergio@gmail.com';
        $recipients[] = 'sochoa@openkm.com';
        $this->ws->sendMail('Testing sending mail from OpenKM', 'Test message body');
        echo 'send Mail';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleMail = new ExampleMail();
$exampleMail->testSendMail();
?>
```

Note samples

Basics

On most methods you'll see 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 -> "11f645a3-281e-4fa6-a2a1-6c1fce5c8ff6";
- Using path -> "/okm:root/SDK4PHP/logo.png"

Methods

addNote

Description:

Method	Return values	Description
addNote(\$nodeId, \$text)	Note	Adds note to a node and returns an object Note.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record</p> <p>\$text string type is the text</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
```

```

    }

    public function testAddNote() {
        try {
            $note = $this->ws->addNote("/okm:root/SDK4PHP/logo.png", "the note text");
            var_dump($note);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testAddNote();

?>

```

getNode

Description:

Method	Return values	Description
getNode(\$noteId)	Note	Retrieves the note.

Parameters:

\$noteId string type

i The noteId is an UUID.

The object Node have a variable named path, in that case the path contains an UUID.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO

```

```

    }

    public function testGetNote() {
        try {
            $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.png");
            if (count($notes) > 0) {
                var_dump($this->ws->getNote($notes[0]->getPath()));
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testGetNote();

?>

```

deleteNote

Description:

Method	Return values	Description
deleteNote(\$noteId)	Note	Deletes a note.

Parameters:

\$noteId string type

i The noteId is an UUID.

The object Node have a variable named path, in that case the path contains an UUID.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

```

```

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
}

public function testDeleteNote(){
    try {
        $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.png");
        if (count($notes) > 0) {
            $this->ws->deleteNote($notes[0]->getPath());
            echo "deleted";
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testDeleteNote();

?>

```

setNote

Description:

Method	Return values	Description
setNote(\$noteId, \$text)	void	Changes the note text.

Parameters:

\$noteId string type

\$text string type is the text

i The noteId is an UUID.

The object Node has a variable named path, in that case the path contains an UUID.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {

    const HOST = "http://localhost:8080/OpenKM/";

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testSetNote(){
    try {
        $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.png");
        if (count($notes) > 0) {
            $this->ws->setNote($notes[0]->getPath(),"text modified");
            echo "updated";
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testSetNote();

?>

```

listNotes

Description:

Method	Return values	Description
listNotes(\$nodeId)	array	Retrieves a list of all notes of a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record.		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

```

```
private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testListNotes() {
    try {
        $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.png");
        foreach ($notes as $note) {
            var_dump($note);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testListNotes();
?>
```

Property samples

Basics

On most methods you'll see 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 -> "adabdb0f-7ff8-4832-9e43-8bc96fc1c9a5";
- Using path -> "/okm:root/SDK4PHP/logo.png"

Methods

addCategory

Description:

Method	Return values	Description
addCategory(\$nodeId, \$catId)	void	Sets a relation between a category and a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record</p> <p>\$catId string type is the text</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}
```

```

    public function testAddCategory() {
        try {
            $this->ws->addCategory("/okm:root/SDK4PHP/logo.png", "/okm:categories/test");
            echo 'add category';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveCategory();

?>

```

removeCategory

Description:

Method	Return values	Description
removeCategory(\$nodeId, \$catId)	void	Removes a relation between a category and a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record		
\$catId string type is the text		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRemoveCategory() {
        try {
            $this->ws->removeCategory("/okm:root/SDK4PHP/logo.png", "/okm:categories/test");
            echo 'remove category';
        }
    }
}

```

```

        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveCategory();

?>

```

addKeyword

Description:

Method	Return values	Description
addKeyword(\$nodeId, \$keyword)	void	Adds a keyword and a node.

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record

\$keyword string type is the keyword

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two_words" (the character "_" is used for the junction).

 Also we suggest you to add keyword in lower case format, because OpenKM is case sensitive.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

```

```

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testAddKeyword() {
    try {
        $this->ws->addKeyword("/okm:root/SDK4PHP/logo.png", "test");
        echo 'add keyword';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testAddKeyword();

?>

```

removeKeyword

Description:

Method	Return values	Description
removeKeyword(\$nodeId, \$keyword)	void	Removes a keyword from a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record</p> <p>\$keyword string type is the keyword</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRemoveKeyword() {

```

```

        try {
            $this->ws->removeKeyword("/okm:root/SDK4PHP/logo.png", "test");
            echo 'remove keyword';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveKeyword();

?>

```

setEncryption

Description:

Method	Return values	Description
setEncryption(\$nodeId, \$cipherName)	void	Marks a document as an encrypted binary data into the repository
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document</p> <p>\$cipherName string type is the cipher name saves information about the encryption mechanism.</p> <div style="border: 1px dashed orange; padding: 5px; margin-top: 10px;">  This method does not perform any kind of encryption, simply mark into the database that a document is encrypted. </div>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```

        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetEncryption() {
        try {
            $this->ws->setEncryption("/okm:root/SDK4PHP/logo.png", "pharase");
            echo 'Set Encryption';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testSetEncryption();
?>

```

unsetEncryption

Description:

Method	Return values	Description
unsetEncryption(\$nodeId)	void	Marks a document as a normal binary data into repository.

Parameters:

\$nodeId string type is the uuid or path of the document



This method does not perform any kind of unryption, simply mark into the database that a document has been unrypted.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}

```

```

    public function testUnsetEncryption() {
        try {
            $this->ws->unsetEncryption("/okm:root/SDK4PHP/logo.png");
            echo 'unset Encryption';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testUnsetEncryption();
?>

```

setSigned

Description:

Method	Return values	Description
setSigned(\$nodeId, \$signed)	void	Marks a document as signed or unsigned binary data into the repository
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document</p> <p>\$signed bool type</p> <div style="border: 1px dashed orange; padding: 5px; margin-top: 10px;">  This method does not perform any kind of digital signature process, simply mark into the database that a document is signed. </div>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetSigned() {
        try {
            $this->ws->setSigned("/okm:root/SDK4PHP/logo.png", true);
            echo 'set Signed';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testSetSigned();
?>
```

PropertyGroup samples

Basics



From older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".

Although we understand this name not helps a lot to identifying these methods with metadata ones, for historical reason, we continue maintaining the nomenclature.

For more information about [Metadata](#).

On almost methods you'll see 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**";
- Using path -> "**/okm:root/SDK4PHP/logo.png**"

Methods

addGroup

Description:

Method	Return values	Description
addGroup(\$nodeId, \$grpName)	void	Adds an empty metadata group to a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record.		
\$grpName string type is the grpName should be a valid Metadata group name.		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {
```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testAddGroup() {
    try {
        $this->ws->addGroup('/okm:root/SDK4PHP/logo.png', 'okg:consulting');
        echo 'addGroup';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testAddGroup();
?>

```

removeGroup

Description:

Method	Return values	Description
removeGroup(\$nodeId, \$grpName)	void	Removes a metadata group of a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record.</p> <p>\$grpName string type is the grpName should be a valid Metadata group name.</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

```

```

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testRemoveGroup() {
    try {
        $this->ws->removeGroup('/okm:root/SDK4PHP/logo.png', 'okg:consulting');
        echo 'Remove Group';
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testRemoveGroup();
?>

```

getGroups

Description:

Method	Return values	Description
getGroups(\$nodeId)	array	Retrieves a list of metadata groups assigned to a node.
Parameters:		
\$nodeId string type is the uuid or path of the document, folder, mail or record.		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetGroups() {
        try {
            $propertyGroups = $this->ws->getGroups('/okm:root/SDK4PHP/logo.png');
            foreach ($propertyGroups as $propertyGroup) {

```

```

        var_dump($propertyGroup);
    }
} catch (Exception $e) {
    var_dump($e);
}
}

}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetGroups();
?>

```

getAllGroups

Description:

Method	Return values	Description
getAllGroups()	array	Retrieves a list of all metadata groups set into the application.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetAllGroups() {
        try {
            $propertyGroups = $this->ws->getAllGroups();
            foreach ($propertyGroups as $propertyGroup) {
                var_dump($propertyGroup);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetAllGroups();
?>

```

getPropertyGroupProperties

Description:

Method	Return values	Description
getPropertyGroupProperties(String nodeId, String grpName)	List<FormElement>	Retrieves a list of all metadata group elements and its values of a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record.</p> <p>\$grpName string type is the grpName should be a valid Metadata group name.</p> <div style="border: 1px dashed #00aaff; border-radius: 10px; padding: 10px; margin-top: 10px;">  The method is usually used to display form elements with its values to be shown or changed by user. </div>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetPropertyGroupProperties() {
        try {
            $formElements = $this->ws->getPropertyGroupProperties('/okm:root/SDK4PHP/');
            foreach ($formElements as $formElement) {
                var_dump($formElement);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
```

```
$examplePropertyGroup->testGetPropertyGroupProperties();
?>
```

getPropertyGroupPropertiesSimple

Description:

Method	Return values	Description
getPropertyGroupPropertiesSimple(\$nodeId, \$grpName)	array	Retrieves a list of all metadata group elements and its values of a node.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record.</p> <p>\$grpName string type is the grpName should be a valid Metadata group name.</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetPropertyGroupPropertiesSimple() {
        try {
            $properties = $this->ws->getPropertyGroupPropertiesSimple('/okm:root/SDK4');
            foreach ($properties as $property) {
                var_dump($property);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetPropertyGroupPropertiesSimple();
?>
```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm(\$grpName)	array	Retrieves a list of all metadata group elements definition.

Parameters:

\$grpName string type is the grpName should be a valid Metadata group name.



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetPropertyGroupForm() {
        try {
            $formElements = $this->ws->getPropertyGroupForm('okg:consulting');
            foreach ($formElements as $formElement) {
                var_dump($formElement);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetPropertyGroupPropertiesSimple();
?>
```

setPropertyGroupProperties

Description:

Method	Return values	Description
setPropertyGroupProperties(\$nodeId, \$grpName, \$formElements)	void	Changes the metadata group

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$grpName string type is the grpName should be a valid Metadata group name.

\$formElements array type is an array of the FormElement



Before changing metadata, you **should have the group added in the node** (see addGroup method) otherwise will error.



Is not mandatory set into parameter ofeList all FormElement, is enough with the formElements 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"
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testSetPropertyGroupProperties() {
    try {
        // Same modification with only affected FormElement
        $formElements = array();
        $name = new \openkm\bean\form\Input();
        $name->setName("okp:consulting.name");
        $name->setValue("new value");
        $formElements[] = $name;
        $this->ws->setPropertyGroupProperties('/okm:root/SDK4PHP/logo.png', 'okp:consulting.name', $formElements);
        echo 'updated';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testSetPropertyGroupProperties();
?>

```

setPropertyGroupPropertiesSimple

Description:

Method	Return values	Description
setPropertyGroupPropertiesSimple(\$nodeId, \$grpName, \$properties)	void	Changes the metadata group

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$grpName string type is the grpName should be a valid Metadata group name.

\$properties array type is an array

 Before changing metadata, you **should have the group added in the node** (see addGroup method) otherwise will error.

 Is not mandatory set into properties parameter all fields values, is enough with the fields you wish to change its value

 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"
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>

```

Example:

```

<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetPropertyGroupPropertiesSimple() {
        try {
            $properties = [];
            $properties["okp:consulting.name"] = "new value";
            $properties["okp:consulting.important"] = "true";

            $this->ws->setPropertyGroupPropertiesSimple('/okm:root/SDK4PHP/logo.png',
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testSetPropertyGroupPropertiesSimple();
?>

```

hasGroup**Description:**

Method	Return values	Description
hasGroup(\$nodeId, \$grpName)	bool	Returns a boolean that indicate if the node has or not a metadata group.
<p>Parameters:</p> <p>\$nodeId string type is the uuid or path of the document, folder, mail or record.</p> <p>\$grpName string type is the grpName should be a valid Metadata group name.</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetPropertyGroupPropertiesSimple() {
        try {
            echo 'Have metadata group: ' . $this->ws->hasGroup('/okm:root/SDK4PHP/log');
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testSetPropertyGroupPropertiesSimple();
?>
```

getRegisteredDefinition

Description:

Method	Return values	Description

getRegisteredDefinition()	string	Return the XML Metada groups definition.
----------------------------------	---------------	--



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRegisteredDefinition() {
        try {
            var_dump($this->ws->getRegisteredDefinition());
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetRegisteredDefinition();
?>
```

getSuggestions

Description:

Method	Return values	Description
getSuggestions(\$nodeId, \$grpName, \$propName)	List<String>	Retrieves a list of a suggested metadata f

Parameters:

\$nodeId string type is the uuid or path of the document, folder, mail or record.

\$grpName string type is the grpName should be a valid Metadata group name.

\$propName string type is the propName parameter should be a [Metadata Select field](#) type.



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



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



The sample below is based on this Metadata group definition:

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetSuggestions() {
        try {
```

```

        $suggestions = $this->ws->getSuggestions('/okm:root/SDK4PHP/logo.png', 'o
        foreach ($suggestions as $suggestion) {
            var_dump($suggestion);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testGetSuggestions();
?>

```

registerDefinition

Description:

Method	Return values	Description
registerDefinition(\$content)	void	Sets the XML Metada groups definition into the repository.

Parameters:

\$content string type is recommend using file_get_contents — Reads entire file into a string

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

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testRegisterDefinition() {
        try {
            $fileName = dirname(__FILE__) . '/propertygroup/propertyGroups.xml';
            $this->ws->registerDefinition(file_get_contents($fileName));
        }
    }
}

```

```
        var_dump('Register Definition correct');
    } catch (Exception $e) {
        var_dump($e);
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testRegisterDefinition();
?>
```

Record samples

On most methods you'll see parameter named "**recId**". The value of this parameter can be a valid record **UUID** or **path**.



Example of recId:

- Using UUID -> "**8e03c11a-fd67-4ba0-8924-180ce538e586**";
- Using path -> "**/okm:root/SDK4PHP/PKI-100200**"

Methods

createRecord

Description:

Method	Return values	Description
createRecord(Record \$record)	Record	Creates a new record and return as a result an object Record.

Parameters:

\$document Record type is an Object Record

The variable **path** into the parameter **record**, must be initialized. It indicates the folder path into OpenKM.

```
Record record = new Record();
record.setPath("/okm:root/PKI-100200");
```

Optionally a title variable can be set, the other variables of the Record (record) will not take any effect on record creation.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
}

public function testCreateRecord() {
    try {
        $record = new Record();
        $record->setPath("/okm:root/SDK4PHP/PKI-100200");
        $record->setTitle("some title");
        $this->ws->createRecord($record);
        var_dump($record);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testCreateRecord();
?>

```

getRecordProperties

Description:

Method	Return values	Description
getRecordProperties(\$recId)	Record	Returns the record properties.
Parameters:		
\$recId string type is the uuid or path of the Record		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }
}

```

```

    public function testGetRecordProperties() {
        try {
            $record = $this->ws->getRecordProperties("/okm:root/SDK4PHP/PKI-100200");
            var_dump($record);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testGetRecordProperties();
?>

```

deleteRecord

Description:

Method	Return values	Description
deleteRecord(\$recId)	void	Deletes a record.
Parameters:		
\$recId string type is the uuid or path of the Record		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO:
    }

    public function testDeleteRecord() {
        try {
            $this->ws->deleteRecord("/okm:root/SDK4PHP/PKI-100200");
            echo 'Deleted';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testDeleteRecord();
?>

```

purgeRecord

Description:

Method	Return values	Description
purgeRecord(\$recId)	void	Records is definitely removed from repository.

Parameters:

\$recId string type is the uuid or path of the Record

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

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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testPurgeRecord() {
        try {
            $this->ws->purgeRecord("/okm:trash/okmAdmin/PKI-100200");
            echo 'Purge';
        } catch (Exception $e) {

```

```

        var_dump($e);
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testPurgeRecord();
?>

```

renameRecord

Description:

Method	Return values	Description
renameRecord(\$recId, \$newName)	Record	Changes the name of a Record and returns the Record

Parameters:

\$recId string type is the uuid or path of the Record

\$newName string type is the new name for the Record

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRenameRecord() {
        try {
            $record = $this->ws->renameRecord("/okm:root/SDK4PHP/PKI-100200", 'new_name');
            var_dump($record);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testRenameRecord();
?>

```

moveRecord

Description:

Method	Return values	Description
moveRecord(\$recId, \$dstId)	void	Moves a record into a folder or record.

Parameters:

\$recId string type is the uuid or path of the Record

\$dstId string type is the uuid or path of the Folder or Record

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testMoveRecord() {
        try {
            $this->ws->moveRecord("/okm:root/SDK4PHP/PKI-100200", "/okm:root/SDK4PHP/");
            echo 'Move';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testMoveRecord();
?>

```

copyRecord

Description:

Method	Return values	Description
copyRecord(\$recId, \$dstId, \$newName)	void	Copies a record into a folder or record.

Parameters:

\$recId string type is the uuid or path of the Record

\$dstId string type is the uuid or path of the Folder or Record

\$newName string type is the new name for the Record. Value is null, record will preserve the same name.

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testcopyRecord() {
        try {
            $this->ws->copyRecord("/okm:root/SDK4PHP/PKI-100200", "/okm:root/SDK4PHP/");
            echo 'Copy';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testcopyRecord();
?>
```

isValidRecord

Description:

Method	Return values	Description
isValidRecord(\$recId)	bool	Returns a boolean that indicates if the node is a record or not.
Parameters:		
\$recId string type is the uuid or path of the Record		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testIsValidRecord() {
        try {
            echo "Is a record:" . $this->ws->isValidRecord("/okm:root/SDK4PHP/PKI-100");
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testIsValidRecord();
?>
```

getRecordChildren

Description:

Method	Return values	Description

getRecordChildren(\$fldId)	array	Returns a list of all records which their parent is fldId
Parameters:		
\$fldId string type is the uuid or path of the Folder or a record node.		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRecordChildren() {
        try {
            $records = $this->ws->getRecordChildren('/okm:root/SDK4PHP');
            foreach ($records as $record) {
                var_dump($record);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testGetRecordChildren();
?>
```

lockRecord

Description:

Method	Return values	Description
lockRecord(\$recId)	LockInfo	Locks a record and returns an object with the Lock information
Parameters:		
\$recId string type is the uuid or path of the Record		



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

A locked record can not be modified by other users.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testLockRecord() {
        try {
            $lockinfo = $this->ws->lockRecord('/okm:root/SDK4PHP/PKI-100200');
            var_dump($lockinfo);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testLockRecord();
?>
```

unlockRecord

Description:

Method	Return values	Description
unlockRecord(\$recId)	void	Unlocks a locked record.
Parameters:		
\$recId string type is the uuid or path of the Record		
<p>Only the user who locked the document is allowed to unlock.</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUnlockRecord() {
        try {
            $this->ws->unlockRecord("/okm:root/SDK4PHP/PKI-100200");
            echo 'unlock';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testUnlockRecord();
?>
```

forceUnlockRecord

Description:

Method	Return values	Description
forceUnlockRecord(\$recId)	void	Unlocks a locked record.
<p>Parameters:</p> <p>\$recId string type is the uuid or path of the Record</p> <p>This method allows to unlock any locked record.</p> <p>It is not mandatory to execute this action by the same user who previously executed the checkout lock action.</p>		



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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testForceUnlockRecord() {
        try {
            $this->ws->forceUnlockRecord("/okm:root/SDK4PHP/PKI-100200");
            echo 'forceUnlock';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testForceUnlockRecord();
?>
```

setRecordTitle

Description:

Method	Return values	Description
setRecordTitle(\$recId, \$title)	void	Sets record title.
Parameters:		
\$recId string type is the uuid or path of the Record		
\$title string type is the title of the Record		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetRecordTitle() {
        try {
            $this->ws->setRecordTitle("/okm:root/SDK4PHP/PKI-100200",'some title');
            echo 'setTitle';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testSetRecordTitle();
?>
```

getRecordPath

Description:

Method	Return values	Description
getRecordPath(\$uuid)	string	Converts record UUID to record path.
Parameters:		
\$uuid string type is the uuid of the Record		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
```

```
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Record;

class ExampleRecord {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRecordPath() {
        try {
            var_dump($this->ws->getRecordPath("8e03c11a-fd67-4ba0-8924-180ce538e586"));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRecord = new ExampleRecord();
$exampleRecord->testGetRecordPath();
?>
```

Relation samples

Basics

On most methods you'll see 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";
- Using path -> "/okm:root/SDK4PHP/logo.png"

Methods

getRelationTypes

Description:

Method	Return values	Description
getRelationTypes(\$type)	array(RelationType)	Retrieves a list of all relations defined of a type.

Parameters:

\$type string type

Available types values:

- RelationType::BIDIRECTIONAL
- RelationType::PARENT_CHILD
- RelationType::MANY_TO_MANY

 More information at [Relation types](#).

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {
```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetRelationTypes() {
    try {
        $relationTypes = $this->ws->getRelationTypes(\openkm\bean\RelationType::P);
        foreach ($relationTypes as $relationType) {
            var_dump($relationType);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testGetRelationTypes();
?>

```

addRelation

Description:

Method	Return values	Description
addRelation(\$nodeAId, \$nodeBId, long relTypeId)	void	Sets a relation between two nodes.
Parameters:		
\$nodeAId string type is the UUID or path of the document, folder, mail or record		
\$nodeBId string type is the UUID or path of the document, folder, mail or record		
\$relTypeId int type is the id RelationType		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testAddRelation() {
    try {
        $relationTypes = $this->ws->getRelationTypes(\openkm\bean\RelationType::P);
        foreach ($relationTypes as $relationType) {
            // looking for a relation named Invoice of
            if($relationType->getTitleAToB() == 'Invoice of'){
                $this->ws->addRelation('/okm:root/SDK4PHP/invoice.pdf', '/okm:root/');
                echo 'add Relation';
            }
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testAddRelation();
?>

```

deleteRelation

Description:

Method	Return values	Description
deleteRelation(\$relationId)	void	Deletes a relation.
<p>Parameters</p> <p>\$relationId int type</p> <div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4; margin-top: 10px;">  Only when the relation will not be used by any node is able to be deleted, otherwise you'll get an error. </div>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testDeleteRelation() {
    try {
        $relations = $this->ws->getRelations('/okm:root/SDK4PHP/invoice.pdf');
        foreach ($relations as $relation) {
            //looking for a relation named Invoice of
            if ($relation->getRelationTitle() == 'Invoice of') {
                $this->ws->deleteRelation($relation->getId());
                echo 'delete Relation';
            }
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testDeleteRelation();
?>

```

getRelations

Description:

Method	Return values	Description
getRelations(\$nodeId)	array(Relation)	Retrieves a list of all relations of a node.
Parameters:		
\$nodeId string type is the UUID or path of the document, folder, mail or record		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetRelations() {
    try {
        $relations = $this->ws->getRelations('/okm:root/SDK4PHP/invoice.pdf');
        foreach ($relations as $relation) {
            var_dump($relation);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testGetRelations();
?>

```

getRelationGroups

Description:

Method	Return values	Description
getRelationGroups(\$nodeId)	array(RelationGroup)	Retrieves a list of all relation groups of a node.
<p>Parameters:</p> <p>\$nodeId string type is the UUID or path of the document, folder, mail or record</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```

        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWO
    }

    public function testGetRelationGroups() {
        try {
            $relationGroups = $this->ws->getRelationGroups('/okm:root/SDK4PHP/invoice
            foreach ($relationGroups as $relationGroup) {
                var_dump($relationGroup);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testGetRelationGroups();
?>

```

addRelationGroup

Description:

Method	Return values	Description
addRelationGroup(\$nodeId, \$groupName, \$type)	void	Adds a relation group at a node.

Parameters:

\$nodeId string type is the **UUID** or **path** of the document, folder, mail or record

\$groupName string type

\$type int type is the

On a relation group only has sense to apply a relation type of Relation Type::MANY_TO_MANY.

 More information at [Relation types](#).

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";

```

```

const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testAddRelationGroup() {
    try {
        $relationTypes = $this->ws->getRelationTypes(\openkm\bean\RelationType::MANY_TO_MANY);
        foreach ($relationTypes as $relationType) {
            if ($relationType->getTitle() == 'staple') {
                $this->ws->addRelationGroup('/okm:root/SDK4PHP/invoice.pdf', 'staple');
                echo 'add Relation Group';
            }
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testAddRelationGroup();
?>

```

addNodeToGroup

Description:

Method	Return values	Description
addNodeToGroup(\$nodeId, \$groupId)	void	Adds a node to an existing relation group.

Parameters:

\$nodeId string type is the **UUID** or **path** of the document, folder, mail or record

\$groupId int type

On a relation group only has sense apply the type RelationType::MANY_TO_MANY.

 More information at [Relation types](#).

Example:

```

<?php
include '../src/openkm/OpenKM.php';

```

```

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testAddNodeToGroup() {
        try {
            $relationGroups = $this->ws->getRelationGroups('/okm:root/SDK4PHP/invoice');
            foreach ($relationGroups as $relationGroup) {
                if ($relationGroup->getName() == 'staple group') {
                    $this->ws->addNodeToGroup('/okm:root/SDK4PHP/complaint.pdf', $relationGroup);
                    echo 'add Node To Group';
                }
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testAddNodeToGroup();
?>

```

deleteRelationGroup

Description:

Method	Return values	Description
deleteRelationGroup(\$nodeId, \$groupId)	void	Removes a node from a relation group.
Parameters:		
\$nodeId string type is the UUID or path of the document, folder, mail or record		
\$groupId int type		

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;

```

```

use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteRelationGroup() {
        try {
            $relationGroups = $this->ws->getRelationGroups('/okm:root/SDK4PHP/invoice');
            foreach ($relationGroups as $relationGroup) {
                if ($relationGroup->getName() == 'staple group') {
                    $this->ws->deleteRelationGroup('/okm:root/SDK4PHP/complaint.pdf',
                    echo 'delete relation group';
                }
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testDeleteRelationGroup();
?>

```

findRelationGroup

Description:

Method	Return values	Description
findRelationGroup(\$groupId)	RelationGroup	Finds a relation group by id.
Parameters:		
\$groupId int type		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;

```

```

use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindRelationGroup() {
        try {
            $groupId = 2;
            var_dump($this->ws->findRelationGroup($groupId));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testFindRelationGroup();
?>

```

setRelationGroupName

Description:

Method	Return values	Description
setRelationGroupName(\$groupId, \$groupName)	void	Changes the relation group name.
Parameters:		
\$groupId int type		
\$groupName string type		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRelation {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";

```

```
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testSetRelationGroupName() {
    try {
        $groupId = 2;
        $this->ws->setRelationGroupName($groupId, 'new name');
        echo 'set group name';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRelation = new ExampleRelation();
$exampleRelation->testSetRelationGroupName();
?>
```

Repository samples

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder()	Folder	Returns the object Folder of node "/okm:root"

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRoolFolder() {
        try {
            $folders = $this->ws->getRootFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetRoolFolder();
?>
```

getTrashFolder

Description:

Method	Return values	Description

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

The returned folder will be the user trash folder.

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetTrashFolder(){
        try {
            $folders = $this->ws->getTrashFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetTrashFolder();
?>
```

getTrashFolderBase

Description:

Method	Return values	Description
getTrashFolderBase()	Folder	Returns the object Folder of node "/okm:trash"

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetTrashFolderBase(){
        try {
            $folders = $this->ws->getTrashFolderBase();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetTrashFolderBase();
?>

```

getTemplatesFolder

Description:

Method	Return values	Description
getTemplatesFolder()	Folder	Returns the object Folder of node "/okm:templates"

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";

```

```

const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetTemplatesFolder() {
    try {
        $folders = $this->ws->getTemplatesFolder();
        var_dump($folders);
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetTemplatesFolder();
?>

```

getPersonalFolder

Description:

Method	Return values	Description
getPersonalFolder()	Folder	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.


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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```

        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetPersonalFolder() {
        try {
            $folders = $this->ws->getPersonalFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetPersonalFolder();
?>

```

getPersonalFolderBase

Description:

Method	Return values	Description
getPersonalFolderBase()	Folder	Returns the object Folder of node "/okm:personal"

Example:

```

package com.openkm;

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

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

        OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);
        try {
            System.out.println(ws.getPersonalFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

getMailFolder

Description:

Method	Return values	Description
getMailFolder()	Folder	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetMailFolder() {
        try {
            $folders = $this->ws->getMailFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetMailFolder();
?>
```

getMailFolderBase

Description:

Method	Return values	Description
getMailFolderBase()	Folder	Returns the object Folder of node "/okm:mail"

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetMailFolderBase(){
        try {
            $folders = $this->ws->getMailFolderBase();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetMailFolderBase();
?>

```

getThesaurusFolder

Description:

Method	Return values	Description
getThesaurusFolder()	Folder	Returns the object Folder of node "/okm:thesaurus"

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetThesaurusFolder() {
    try {
        $folders = $this->ws->getThesaurusFolder();
        var_dump($folders);
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetThesaurusFolder();
?>

```

getCategoriesFolder

Description:

Method	Return values	Description
getCategoriesFolder()	Folder	Returns the object Folder of node "/okm:categories"

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetCategoriesFolder() {
        try {
            $folders = $this->ws->getCategoriesFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetCategoriesFolder();
?>

```

purgeTrash

Description:

Method	Return values	Description
purgeTrash()	void	Definitively remove from repository all nodes into "/okm:trash/{userId}"

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

 When a node is purged only will be able to be restored from a previously repository backup. The purge action remove the node definitively from the repository.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testPurgeTrash() {
        try {
            $this->ws->purgeTrash();
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
}

```

```

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testPurgeTrash();
?>

```

getUpdateMessage

Description:

Method	Return values	Description
getUpdateMessage()	string	Retrieves a message when a new OpenKM release is available.

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



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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetUpdateMessage() {
        try {
            var_dump($this->ws->getUpdateMessage());
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetUpdateMessage();
?>

```

getRepositoryUuid

Description:

Method	Return values	Description
getRepositoryUuid()	string	Retrieves installation unique identifier.

Example:

```

<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetRepositoryUuid() {
        try {
            var_dump($this->ws->getRepositoryUuid());
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetRepositoryUuid();
?>

```

hasNode

Description:

Method	Return values	Description
hasNode(\$nodeId)	bool	Returns a node that indicate if a node exists or not.

Parameters:

\$nodeId string type is the value of the parameter nodeId can be a valid UUID or path.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testHasNode() {
        try {
            echo 'Exists node: ' . $this->ws->hasNode('adabdb0f-7ff8-4832-9e43-8bc96f');
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testHasNode();
?>
```

getNodePath

Description:

Method	Return values	Description
getNodePath(\$uuid)	string	Converts node UUID to path.
Parameters:		
\$uuid string type is the uuid of the node		

Example:

```
<?php
```

```

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetNodePath() {
        try {
            var_dump($this->ws->getNodePath('adabdb0f-7ff8-4832-9e43-8bc96fc1c9a5'));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetNodePath();
?>

```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(\$nodePath)	string	Converts node path to UUID.
Parameters:		
\$nodePath string type is the path of the node		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

```

```

const HOST = "http://localhost:8080/OpenKM/";
const USER = "okmAdmin";
const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testGetNodeUuid() {
    try {
        var_dump($this->ws->getNodeUuid('/okm:root/SDK4PHP/logo.png'));
    } catch (Exception $e) {
        var_dump($e);
    }
}
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetNodeUuid();
?>

```

getAppVersion

Description:

Method	Return values	Description
getAppVersion()	AppVersion	Returns information about OpenKM version.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetAppVersion() {
        try {
            $appVersion = $this->ws->getAppVersion();
            var_dump($appVersion);
        } catch (Exception $e) {

```

```

        var_dump($e);
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetAppVersion();
?>

```

copyAttributes

Description:

Method	Return values	Description
copyAttributes(\$nodeId, \$dstId, \$categories, \$keywords, \$propertyGroups, \$notes, \$wiki)	void	Copies attributes from a node to other.

Parameters:

\$nodeId string type is the path of the node

\$dstId string type is the path of the node

\$categories bool type

\$keywords bool type

\$propertyGroups bool type

\$notes bool type

\$wiki bool type

i

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

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testCopyAttributes() {
        try {
            $this->ws->copyAttributes('/okm:root/SDK4PHP/invoice.pdf', '/okm:root/SDK4PHP/invoice.pdf');
            echo 'Correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testCopyAttributes();
?>
```

executeScript

Description:

Method	Return values	Description
executeScript(\$content)	ScriptExecutionResult	Executes an script.

Parameters:

\$content string type Recommends using file_get_contents — Reads entire file into a string

The local script - test.bsh - used in the sample below:

```
import com.openkm.bean.*;
import com.openkm.api.*;

for (Folder fld : OKMFolder.getInstance().getChildren(null, "/okm:root")) {
```

```

    print(fld+"\n");
}
// Some value can also be returned as string
return "some result";

```



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

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testExecuteScript() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.bsh';
            $scriptExecutionResult = new \openkm\bean\ScriptExecutionResult();
            $scriptExecutionResult = $this->ws->executeScript(file_get_contents($fileName));
            var_dump($scriptExecutionResult->getResult());
            var_dump($scriptExecutionResult->getStdout());
            if ($scriptExecutionResult->getStderr() != '') {
                echo "Error happened";
                var_dump($scriptExecutionResult->getStderr());
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testExecuteScript();
?>

```

executeSqlQuery

Description:

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

executeSqlQuery(\$content)	SqlQueryResults	Executes SQL sentences.
<p>Parameters:</p> <p>\$content string type Recommend using <code>file_get_contents</code> — Reads entire file into a string</p> <p>The test.sql script used in the sample below:</p> <pre style="border: 1px dashed blue; padding: 5px;">SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;</pre> <div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;"> <p> The SQL script can only contains a single SQL sentence.</p> <p>This action can only be done by a super user (user with ROLE_ADMIN).</p> </div>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;
use openkm\bean\SqlQueryResults;
use openkm\bean\SqlQueryResultColumns;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testExecuteSqlQuery() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.sql';
            $sqlQueryResults = new SqlQueryResults();
            $sqlQueryResults = $this->ws->executeSqlQuery(file_get_contents($fileName));
            foreach ($sqlQueryResults->getResults() as $sqlQueryResultColumns) {
                $columns = $sqlQueryResultColumns->getColumns();
                var_dump('uuid: ' . $columns[0] . ' name: ' . $columns[1]);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
```

```
$exampleRepository = new ExampleRepository();
$exampleRepository->testExecuteSqlQuery();
?>
```

Also the `InputStream` can be set as:

```
$sql = "SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;";
$sqlQueryResults = $this->ws->executeSqlQuery($sql);
```

executeHqlQuery

Description:

Method	Return values	Description
executeHqlQuery(\$content)	HqlQueryResults	Executes HQL sentences.

Parameters:

\$content string type Recommend using `file_get_contents` — Reads entire file into a string

The `testhql.sql` script used in the sample below:

```
SELECT uuid, name from NodeBase where name = 'okm:root';
```

 The HQL script can only contains a single HQL sentence.
 This action can only be done by a super user (user with `ROLE_ADMIN`).

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}
```

```

public function testExecuteHqlQuery() {
    try {
        $fileName = dirname(__FILE__) . '/files/testhql.sql';
        $hqlQueryResults = new \openkm\bean\HqlQueryResults();
        $hqlQueryResults = $this->ws->executeHqlQuery(file_get_contents($fileName));
        foreach ($hqlQueryResults->getResults() as $hqlQueryResult ){
            var_dump($hqlQueryResult);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testExecuteHqlQuery();
?>

```

Also the `InputStream` can be set as:

```

$hql = "SELECT uuid, name from NodeBase where name = 'okm:root'";
$hqlQueryResults = $this->ws->executeHqlQuery($hql);

```

getTranslations

Description:

Method	Return values
getTranslations(\$lang, \$module)	array Retrie

Parameters:

\$lang string type

\$module string type

i The OpenKM translations tables can be used to retrieve actually OpenKM translations or create your own translations.

By default modules values are :

- frontend (used by default OpenKM UI).
- extension (used by OpenKM extension UI).
- mobile (used by OpenKM mobile UI).

Example to add a new Translation module :

SQL values to be executed from [Database query](#) view:

```
DELETE FROM OKM_TRANSLATION WHERE TR_LANGUAGE='en-GB' and TR_MODULE='doc';
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUE:
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUE:
```

The code then should be:

```
$translations = $this->ws->getTranslations('en-GB', 'doc');
```

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;
use openkm\bean\SqlQueryResults;
use openkm\bean\SqlQueryResultColumns;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetTranslations() {
        try {
            $translations = $this->ws->getTranslations('en-GB','doc');
            foreach ($translations as $translation ){
                var_dump('key: ' . $translation->getKey() . ', with translation: ' . $translation->getText());
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetTranslations();
?>
```

getConfiguration

Description:

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

getConfiguration(\$key)	Configuration	Retrieve the value of a configuration parameter.
--------------------------------	----------------------	--

Parameters:

\$key string type is a configuration parameter



If your OpenKM version have the configuration parameter named "**webservices.visible.properties**", will be restricted for non Administrator users what parameters are accessible. That means any non Administrator use who will try accessing across the webservices to configuration parameters not set into the list of values of "**webservices.visible.properties**" will get an access denied exception.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;
use openkm\bean\SqlQueryResults;
use openkm\bean\SqlQueryResultColumns;

class ExampleRepository {

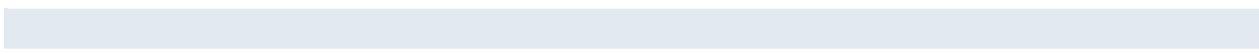
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetConfiguration(){
        try {
            $configuration = $this->ws->getConfiguration('system.ocr');
            var_dump($configuration);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetConfiguration();
?>
```

getChangeLog**Description:**

Method	Return values	Description
getChangeLog(\$nodePath, \$modificationsFrom)	array(ChangeLogged)	Return the list of changes in some path and subfolders.
<p>Parameters:</p> <p>\$nodePath string type is the path of the node(Folder,Document, Mail, Record)</p> <p>\$modificationsFrom string type is the modificacions from date</p>		
 <ul style="list-style-type: none"> The method is used by desktop synchronization application for retrieving the changes. 		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;
use openkm\bean\SqlQueryResults;
use openkm\bean\SqlQueryResultColumns;

class ExampleRepository {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetChangeLog() {
        try {
            $changeLoggedes = $this->ws->getChangeLog("/okm:root/SDK4PHP", date('Y-m-d'));
            foreach ($changeLoggedes as $changeLogged) {
                var_dump($changeLogged);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetChangeLog();
?>
```

Search samples

Basics

Most methods use QueryParams here there're some clues about how using it.

Variables	MType	Allow wildcards	Restrictions
domain	int	No.	<p>Available values:</p> <ul style="list-style-type: none"> • QueryParams::DOCUMENT • QueryParams::FOLDER • QueryParams::MAIL • QueryParams::RECORD <p>By default the value is set to QueryParams.DOCUMENT.</p> <p>For searching documents and folders use value:</p> <pre>(QueryParams::DOCUMENT QueryParams::FOLDER)</pre>
author	string	No.	Value must be a valid userId.
name	string	Yes.	
title	string	Yes.	
keywords	array	Yes.	
categories	array	No.	Values should be categories UUID, not use path value.

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

mailFrom	string	Yes.	Only apply to mail nodes.
mailTo		Yes.	Only apply to mail nodes.
notes		Yes.	
properties	array	Yes on almost.	<p>On metadata field values like "date" can not be applied wilcards.</p> <p>The map of the properties is composed of pairs: (<code>metadata_field_name</code>,<code>metada_field_value</code>)</p> <p>For example:</p> <pre style="border: 1px dashed blue; padding: 5px;">\$properties = array(); \$properties['okp:consulting.name'] = 'name value';</pre>

 The search operation is done only by AND logic.

Wildcard examples:

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

Methods

findByContent

Description:

Method	Return values	Description
findByContent(\$content)	array	Returns a list of QueryResults filtered by the value of the content parameter.

Parameters:

\$content string type

 The method only searches among all documents, it not takes in consideration any other kind of nodes.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindByContent() {
        try {
            $queryResults = $this->ws->findByContent('test*');
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
```

```
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindByContent();
?>
```

findByName

Description:

Method	Return values	Description
findByName(\$name)	array	Returns a list of QueryResults filtered by the value of the name parameter.

Parameters:

\$name string type

 The method only searches among all documents, it not takes in consideration any other kind of nodes.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindByName() {
        try {
            $queryResults = $this->ws->findByName('test');
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindByName();
?>
```

findByKeywords

Description:

Method	Return values	Description
findByKeywords(\$keywords)	array	Returns a list of QueryResults filtered by the values of the keywords parameter.

Parameters:

\$keywords array type

 The method only searches among all documents, it not takes in consideration any other kind of nodes.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindByKeywords(){
        try {
            $keywords = array();
            $keywords[] = 'php';
            $queryResults = $this->ws->findByKeywords($keywords);
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindByKeywords();
```

```
?>
```

find

Description:

Method	Return values	Description
find(QueryParams \$queryParams, \$propertiesPlugin)	array	Returns a list of QueryResults filtered by the values of the queryParams parameter.

Parameters:

\$queryParams QueryParams type

\$propertiesPlugin string type, must be the canonical class name of the class which implements the NodeProperties interface.

i Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFind() {
        try {
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryParams::FOLDER);
            $queryParams->setFolder("398735af-6282-450e-863c-d00390c5bdda");
            $queryParams->setFolderRecursive(true);
            $queryParams->setLastModifiedFrom(20150628000000);
            $queryParams->setLastModifiedTo(date('Ymdhis'));
        }
    }
}
```

```

        $queryResults = $this->ws->find($queryParams, null);
        foreach ($queryResults as $queryResult) {
            var_dump($queryResult);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$search = new ExampleSearch();
$search->testFind();
?>

```

findPaginated

Description:

Method	Return values	Description
findPaginated(QueryParams \$queryParams, \$offset, \$limit, \$propertiesPlugin)	ResultSet	Returns a list of paginated results filtered by the values of the queryParams parameter.
<p>Parameters:</p> <p>\$queryParams QueryParams type</p> <p>\$offset int type</p> <p>\$limit int type</p> <p>\$propertiesPlugin string type, must be the canonical class name of the class which implements the NodeProperties interface.</p> <div style="border: 1px dashed green; padding: 10px; margin: 10px 0;"> <p> The parameter "limit" and "offset" allow you to retrieve just a portion of the results of a query.</p> <ul style="list-style-type: none"> • The parameter "limit" is used to limit the number of results returned. • The parameter "offset" says to skip that many results before the beginning to return results. </div> <div style="border: 1px dashed blue; padding: 10px; margin: 10px 0;"> <p> For example if your query have 1000 results, but you only want to return the first 10, you should use these values:</p> <ul style="list-style-type: none"> • limit=10 • offset=0 <p>Now suppose you want to show the results from 11-20, you should use these values:</p> </div>		

- limit=10
- offset=10

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindPaginated(){
        try {
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryParams::FOLDER);
            $queryParams->setFolder("398735af-6282-450e-863c-d00390c5bdda");
            $queryParams->setFolderRecursive(true);
            $queryParams->setLastModifiedFrom(20150628000000);
            $queryParams->setLastModifiedTo(date('Ymdhis'));
            $resultSet = $this->ws->findPaginated($queryParams,0,10, null);
            echo "Total results:" . $resultSet->getTotal();
            foreach ($resultSet->getResults() as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindPaginated();
?>
```

findByQuery

Description:

Return

Method	values	Description
findByQuery(\$query, \$propertiesPlugin)	array	Returns a list of paginated results filtered by the values of the query parameter.

Parameters:

\$query string type

\$propertiesPlugin string type, must be the canonical class name of the class which implements the NodeProperties interface.

i The **syntax** to use in the **query** 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:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindByQuery() {
        try {
            $queryResults = $this->ws->findByQuery('keyword:invoice AND title:document');
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```

```

$openkm = new OpenKM(); //autoload
$search = new ExampleSearch();
$search->testFindByQuery();
?>
    
```

findByQueryPaginated

Description:

Method	Return values	Description
findByQueryPaginated(\$query, \$offset, \$limit, \$propertiesPlugin)	ResultSet	Returns a list of paginated results filtered by the values of the query parameter.

Parameters:

\$query string type

\$offset int type

\$limit int type

\$propertiesPlugin string type, must be the canonical class name of the class which implements the NodeProperties interface.

i The **syntax** to use in the **query** parameter is the pair '**field:value**'. For example:

- "name:grial" is filtering field name by word grial.

More information about Lucene sintaxis at [Lucene query syntax](#).

✓ The parameter "limit" and "offset" 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.

i For example if your query have 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:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindByQueryPaginated() {
        try {
            $resultSet = $this->ws->findByQueryPaginated('keyword:invoice AND title:document');
            echo "Total results:" . $resultSet->getTotal();
            foreach ($resultSet->getResults() as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindByQueryPaginated();
?>
```

findSimpleQueryPaginated

Description:

Method	Return values	Description
findSimpleQueryPaginated(\$statement, \$offset, \$limit)	ResultSet	Returns a list of paginated results filtered by the values of the statement parameter.

Parameters:**\$statement** string type**\$offset** int type**\$limit** int type

The **syntax** to use in the **statement** parameter is the pair '**field:value**'. For example:

- "name:grial" is filtering field name by word grial.

More information about Lucene sintaxis at [Lucene query syntax](#).



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

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



For example if your query have 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:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }
}
```

```

    }

    public function testFindSimpleQueryPaginated() {
        try {
            $resultSet = $this->ws->findSimpleQueryPaginated('name:grial',0,10);
            echo "Total results:" . $resultSet->getTotal();
            foreach ($resultSet->getResults() as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$search = new ExampleSearch();
$search->testFindSimpleQueryPaginated();
?>

```

findMoreLikeThis

Description:

Method	Return values	Description
findMoreLikeThis(String uuid, int max)	ResultSet	Returns a list of documents that are considered similar by search engine.

Parameters:

\$statement string type is the uuid of the Document

\$offset int type is the max value is used to limit the number of results returned.

 The method can only be used with documents.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";
}

```

```

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testFindMoreLikeThis(){
    try {
        $resultSet = $this->ws->findMoreLikeThis("96c44de6-1d0d-45fb-b380-4984f461");
        echo "Total results:" . $resultSet->getTotal();
        foreach ($resultSet->getResults() as $queryResult) {
            var_dump($queryResult);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindMoreLikeThis();
?>

```

getKeywordMap

Description:

Method	Return values	Description
getKeywordMap(\$filter)	array	Returns a array of the KeywordMap with its count value filtered by other keywords.

Parameters:

\$filter array type is the uuid of the Document

i Example:

- Doc1.txt has keywords "test", "one", "two".
- Doc2.txt has keywords "test", "one"
- Doc3.txt has keywords "test", "three".

The results filtering by "test" -> "one", "two", "three".

The results filtering by "one" -> "test", "two".

The results filtering by "two" -> "test", "one".

The results filtering by "three" -> "test".

The results filtering by "one" and "two" -> "test".

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testKeywordMap(){
        try {
            // All keywords without filtering
            echo 'Without filtering';
            $keywordMaps = $this->ws->getKeywordMap();
            foreach ($keywordMaps as $keywordMap) {
                var_dump($keywordMap);
            }
            // Keywords filtered
            echo 'Filtering';
            $filter = array('test', 'php');
            $keywordMaps = $this->ws->getKeywordMap($filter);
            foreach ($keywordMaps as $keywordMap) {
                var_dump($keywordMap);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testKeywordMap();
?>
```

getCategorizedDocuments

Description:

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

Parameters:

\$categoryId string type is the uuid or path of the Category

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetCategorizedDocuments(){
        try {
            $documents = $this->ws->getCategorizedDocuments('abd631c5-93b8-4265-98f2-...');
            foreach ($documents as $document) {
                var_dump($document);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testGetCategorizedDocuments();
?>
```

saveSearch

Method	Return values	Description
saveSearch(QueryParams \$params)	int	Saves a search parameters and returns the id of the saved search

Parameters:

\$params QueryParams type is the variable queryName of the parameter params, should have to be initialized.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSaveSearch() {
        try {
            $params = new QueryParams();
            $params->setDomain(QueryParams::DOCUMENT + QueryParams::FOLDER);
            $params->setName('test*');
            $params->setFolder("398735af-6282-450e-863c-d00390c5bdda");
            $params->setFolderRecursive(true);
            $params->setLastModifiedFrom(20150628000000);
            $params->setLastModifiedTo(date('Ymdhis'));
            $queryResults = $this->ws->find($params, null);
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
            $params->setQueryName('sample search');
            var_dump($this->ws->saveSearch($params));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testSaveSearch();
?>
```

updateSearch

Description:

Method	Return values	Description
updateSearch(QueryParams \$params)	void	Updates a previously saved search parameters.
Parameters:		
\$params QueryParams type		



Only a saved search created by the same user who's executing the method can be updated.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testUpdateSearch() {
        try {
            $qpId = 1; // Some valid search id
            $params = $this->ws->getSearch($qpId);
            $params->setName('test*.pdf');
            $this->ws->updateSearch($params);
            echo 'update search';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testUpdateSearch();
?>
```

getSearch

Description:

Method	Return values	Description
getSearch(\$qpId)	QueryParams	Gets a saved search parameters.
Parameters:		
\$qpId int type is the id of the saved search		



Only can be retrieved a saved search created by the same user who's executing the method.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetSearch() {
        try {
            $qpId = 2; // Some valid search id
            $params = $this->ws->getSearch($qpId);
            var_dump($params);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testGetSearch();
?>
```

getAllSearchs

Description:

Method	Return values	Description
getAllSearchs()	List<QueryParams>	Retrieves a list of all saved search parameters.



Only will be retrieved the list of the saved searches created by the same user who's executing the method.

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetAllSearchs() {
        try {
            foreach ($this->ws->getAllSearchs() as $params) {
                var_dump($params);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testGetAllSearchs();
?>

```

deleteSearch

Description:

Method	Return values	Description
deleteSearch(int qpId)	void	Deletes a saved search parameters.
Parameters:		
\$qpId int type is the id of the saved search		
 Only can be deleted a saved search created by the same user user who's executing the method.		

Example:

```

<?php

```

```
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\QueryParams;

class ExampleSearch {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteSearch() {
        try {
            $qpId = 2; // Some valid search id
            $this->ws->deleteSearch($qpId);
            echo 'delete search';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testGetAllSearchs();
?>
```

Workflow samples

For almost all examples has been used the [Purchase workflow sample](#).

Methods

registerProcessDefinition

Description:

Method	Return values	Description
registerProcessDefinition(\$content)	void	Register a new workflow.
Parameters:		
\$content string type is recommend using <code>file_get_contents</code> — Reads entire file into a string		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRegisterProcessDefinition() {
        try {
            $fileName = dirname(__FILE__) . '/workflow/Purchase.par';
            $this->ws->registerProcessDefinition(file_get_contents($fileName));
            var_dump('Register Definition correct');
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```

```

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testRegisterProcessDefinition();
?>

```

deleteProcessDefinition

Description:

Method	Return values	Description
deleteProcessDefinition(\$pdId)	void	Deletes a workflow.
<p>Parameters:</p> <p>\$pdId integer type is a valid workflow process definition.</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testDeleteProcessDefinition() {
        try {
            $pdId = 11; // Valid workflow process definition
            $this->ws->deleteProcessDefinition($pdId);
            var_dump('Delete Process Definition correct');
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testDeleteProcessDefinition();
?>

```

getProcessDefinition

Description:

Method	Return values	Description
getProcessDefinition(\$pdId)	ProcessDefinition	Returns a workflow process definition.
Parameters:		
\$pdId integer type is a valid workflow process definition.		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetProcessDefinition() {
        try {
            $pdId = 83; // Valid workflow process definition
            $processDefinition = $this->ws->getProcessDefinition($pdId);
            var_dump($processDefinition);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testGetProcessDefinition();
?>
```

runProcessDefinition

Description:

Method	Return values	Description
runProcessDefinition(\$pdId, \$uuid, \$formElements = array)	ProcessInstance	Executes a workflow on some node and Returns a workflow process instance
<p>Parameters:</p> <p>\$pdId int type is a valid workflow process definition.</p> <p>\$uuid string type is the uuid or path of the document, folder, mail or record.</p> <p>\$formElements array type are form element values needed for starting the workflow (not all workflows need form values for starting).</p>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testRunProcessDefinition() {
        try {
            $pdId = 91; // Valid workflow process definition
            // Same modification with only affected FormElement
            $formElements = array();
            $price = new \openkm\bean\form\Input();
            $price->setName("price");
            $price->setValue(1000);
            $formElements[] = $price;

            $textArea = new \openkm\bean\form\TextArea();
            $textArea->setName("description");
            $textArea->setValue("some description here");
            $formElements[] = $textArea;

            $processInstance = $this->ws->runProcessDefinition($pdId, '8e66f95b-90b4-
```

```

        var_dump($processInstance);
    } catch (Exception $e) {
        var_dump($e);
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testRunProcessDefinition();
?>

```

FindAllProcessDefinitions

Description:

Method	Return values	Description
findAllProcessDefinitions()	array(ProcessDefinition)	Retrieves a list of all registered workflows definitions.

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindAllProcessDefinitions() {
        try {
            $processDefinitions = $this->ws->findAllProcessDefinitions();
            foreach ($processDefinitions as $processDefinition) {
                var_dump($processDefinition);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindAllProcessDefinitions();
?>

```

findProcessInstances

Description:

Method	Return values	Description
findProcessInstances(\$pdId)	array(ProcessInstance)	Retrieves a list of all process instances of some registered workflows definition.
Parameters:		
\$pdId int type is a valid workflow process definition.		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

ini_set('display_errors', true);
error_reporting(E_ALL);

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindProcessInstances() {
        try {
            // Get all workflow definitions
            $processDefinitions = $this->ws->findAllProcessDefinitions();
            foreach ($processDefinitions as $processDefinition) {
                echo '<h2>WF definition</h2>';
                var_dump($processDefinition);
            }
            // Get all process of some workflow definition
            $processInstances = $this->ws->findProcessInstances($processDefinition);
            foreach ($processInstances as $processInstance) {
                echo '<h2>Process Instance</h2>';
                var_dump($processInstance);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```
$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindProcessInstances();
?>
```

findLatestProcessDefinitions

Description:

Method	Return values	Description
findLatestProcessDefinitions()	array(ProcessDefinition)	Retrieves a list of the last workflows definitions.


Can be several versions of the same workflow registered.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindLatestProcessDefinitions() {
        try {
            // Get all latest workflow definitions
            $processDefinitions = $this->ws->findLatestProcessDefinitions();
            foreach ($processDefinitions as $processDefinition) {
                var_dump($processDefinition);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindLatestProcessDefinitions();
?>
```

findLastProcessDefinition

Description:

Method	Return values	Description
findLastProcessDefinition(\$name)	ProcessDefinition	Retrieves last workflow definition of some specific workflow.

Parameters:

\$name string type Identify an specific workflow definitions group.

 Several workflow definition versions they have the same name.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindLastProcessDefinition() {
        try {
            $processInstance = $this->ws->findLastProcessDefinition("purchase");
            var_dump($processInstance);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindLastProcessDefinition();
?>
```

getProcessInstance

Description:

Method	Return values	Description
getProcessInstance(\$piId)	ProcessInstance	Returns the process instance.
Parameters:		
\$piId string type is a valid process instance id.		

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetProcessInstance() {
        try {
            $piId = 77; // Some valid process instance id
            $processInstance = $this->ws->getProcessInstance($piId);
            var_dump($processInstance);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testGetProcessInstance();
?>
```

findUserTaskInstances

Description:

Method	Return values	Description
findUserTaskInstances()	array(TaskInstance)	Retrieves a list of task instances assigned to the user.

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindUserTaskInstances() {
        try {
            // Get all user task instances
            $taskInstances = $this->ws->findUserTaskInstances();
            foreach ($taskInstances as $taskInstance) {
                var_dump($taskInstance);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindUserTaskInstances();
?>
```

findTaskInstances

Description:

Method	Return values	Description
findTaskInstances(\$pId)	array(TaskInstance)	Retrieves a list of task instances of some process instance id.
Parameters:		
\$pId string type is a valid process instance id.		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testFindTaskInstances() {
        try {
            // Get all task instances of some process instance
            $piId = 77; // Some valid process instance id
            $taskInstances = $this->ws->findTaskInstances($piId);
            foreach ($taskInstances as $taskInstance) {
                var_dump($taskInstance);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testFindTaskInstances();
?>

```

setTaskInstanceValues

Description:

Method	Return values	Description
setTaskInstanceValues(\$taskId, \$transName, \$formElements = array())	void	Retrieves a list of task instances of some process instance id.
Parameters:		
\$taskId string type is a valid task instance id.		
\$transName string type is the chosen transaction.		
\$formElements array type are form element values needed for starting the workflow (not all workflows need form values		

for starting).

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetTaskInstanceValues() {
        try {
            $stiId = 5; // Some valid task instance id
            $formElements = array();
            $this->ws->setTaskInstanceValues($stiId, 'approve', $formElements);
            echo '<p>Set task Instance value</p>';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testSetTaskInstanceValues();
?>
```

getTaskInstance

Description:

Method	Return values	Description
getTaskInstance(\$stiId)	TaskInstance	Returns a task instance.
Parameters:		
\$stiId string type is a valid task instance id.		

Example:

```
<pre>
```

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetTaskInstance() {
        try {
            $stiId = 282; // Some valid task instance id
            $taskInstance = $this->ws->getTaskInstance($stiId);
            var_dump($taskInstance);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testGetTaskInstance();
?>

```

setTaskInstanceActorId

Description:

Method	Return values	Description
setTaskInstanceActorId(\$stiId, \$actorId)	void	Starts a task instance.
<p>Parameters:</p> <p>\$stiId string type is a valid task instance id.</p> <p>\$actorId string type must be some valid OpenKM userId.</p>		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

```

```

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testSetTaskInstanceActorId() {
        try {
            $tiId = 4; // Some valid task instance id
            $this->ws->setTaskInstanceActorId($tiId, 'okmAdmin');
            echo '<p>Set task Instance actorid</p>';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testSetTaskInstanceActorId();
?>

```

startTaskInstance

Description:

Method	Return values	Description
startTaskInstance(\$tiId)	void	Starts a task instance.
Parameters:		
\$tiId string type is a valid task instance id.		

Example:

```

<?php

include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";

```

```

const PASSWORD = "admin";

private $ws;

public function __construct() {
    $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
}

public function testStartTaskInstance() {
    try {
        $tiId = 4; // Some valid task instance id
        $this->ws->startTaskInstance($tiId);
        echo '<p>Start task Instance</p>';
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testStartTaskInstance();
?>

```

endTaskInstance

Description:

Method	Return values	Description
endTaskInstance(\$tiId, \$transName)	void	Ends a task instance.
<p>Parameters:</p> <p>\$tiId string type is a valid task instance id.</p> <p>\$transName string type is the chosen transaction.</p>		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {

```

```

        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testEndTaskInstance() {
        try {
            $tiId = 6; // Some valid task instance id
            $this->ws->endTaskInstance($tiId, 'end');
            echo '<p>End task Instance</p>';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testEndTaskInstance();
?>

```

getProcessDefinitionForms

Description:

Method	Return values	Description
getProcessDefinitionForms(\$pdId)	array[String,array[formElement]]	Return a map with all the process definition forms.
Parameters:		
\$pdId string type is valid workflow process definition		

Example:

```

<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleWorkflow {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetProcessDefinitionForms() {
        try {

```

```
        $pdId = 3; // Valid workflow process definition
        $processDefinitionForms = $this->ws->getProcessDefinitionForms($pdId);
        foreach ($processDefinitionForms as $key => $value) {
            var_dump('key' . $key);
            var_dump($value);
        }
    } catch (Exception $e) {
        var_dump($e);
    }
}

}

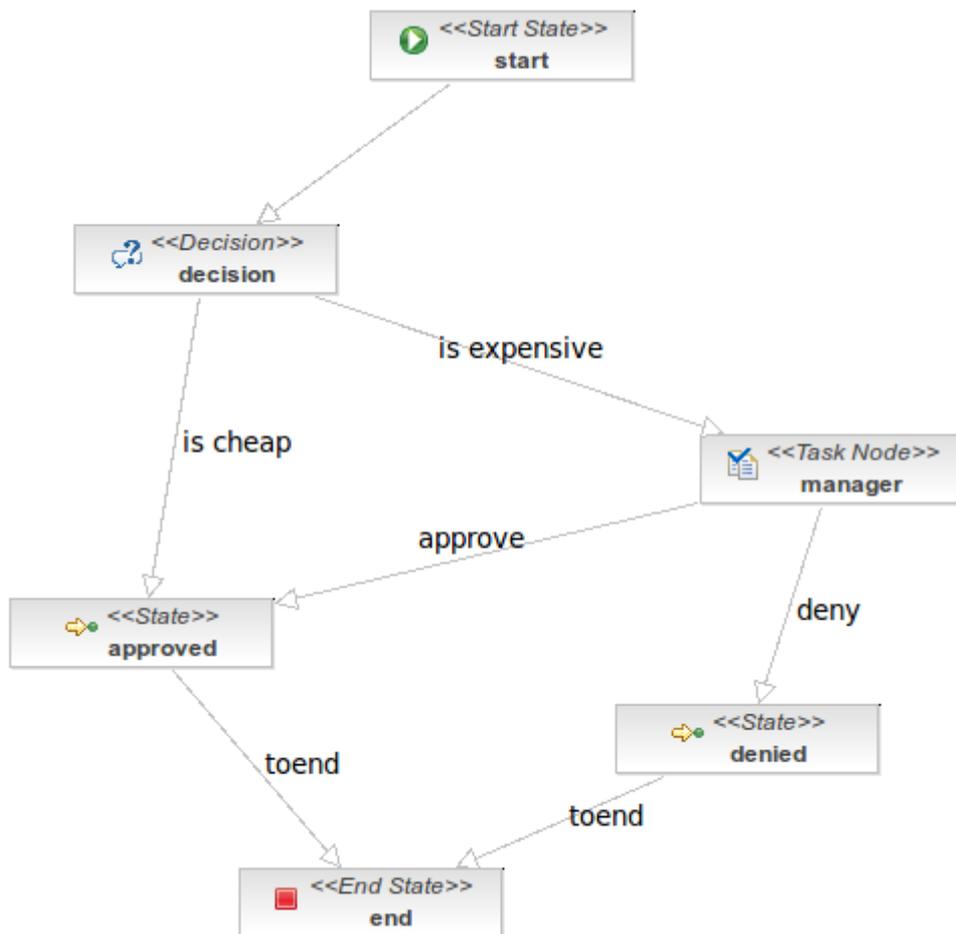
$openkm = new OpenKM(); //autoload
$exampleWorkflow = new ExampleWorkflow();
$exampleWorkflow->testGetProcessDefinitionForms();
?>
```

Purchase workflow sample

The tasks will be assigned to a user called "manager" so you need to create this user and log 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:jbpn.org:jpd1-3.2" 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>

  <task name="manager">
    <assignee>manager</assignee>
  </task>

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

  <state name="denied">
    <toend to="end"></toend>
  </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" read
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>

```

Report samples

Methods

getReports

Description:

Method	Return values	Description
getReports(\$active)	array(Report)	Returns a list of reports.

Parameters:

\$active bool type is the active reports

Example:

```
<?php
include '../src/openkm/OpenKM.php';

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleReport {

    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetReports() {
        try {
            $reports = $this->ws->getReports(true);
            foreach ($reports as $report) {
                var_dump($report);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleReport = new ExampleReport();
$exampleReport->testGetReports();
?>
```

getReport

Description:

Method	Return values	Description
getReport(\$rpId)	Report	Returns a report.
Parameters:		
\$rpId int type is valid report id		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleReport {
    const HOST = "http://localhost:8080/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testGetReport() {
        try {
            var_dump($this->ws->getReport(2));
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$exampleReport = new ExampleReport();
$exampleReport->testGetReport();
?>
```

executeReport

Description:

Method	Return values	Description

executeReport(\$rpId, \$params, \$format)	InputStream	Return a document result of executing a report.
<p>Parameters:</p> <p>\$rpId int type is valid report id</p> <p>\$params array type is array with key and value</p> <p>\$format string type is valid report id</p> <div style="border: 1px dashed #ccc; padding: 10px; margin: 10px 0;"> <p>i Available formats:</p> <ul style="list-style-type: none"> • Report::FORMAT_CSV • Report::FORMAT_DOCX • Report::FORMAT_HTML • Report::FORMAT_ODT • Report::FORMAT_PDF • Report::FORMAT_RTF • Report::FORMAT_TEXT </div>		

Example:

```
<?php
include '../src/openkm/OpenKM.php';
use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleReport {
    const HOST = "http://localhost:8888/OpenKM/";
    const USER = "okmAdmin";
    const PASSWORD = "admin";

    private $ws;

    public function __construct() {
        $this->ws = OKMWebServicesFactory::build(self::HOST, self::USER, self::PASSWORD);
    }

    public function testExecuteReport($method) {
        $params = [];
        $params['from_date'] = '2017-01-01';
        $params['to_date'] = '2017-06-19';
        $format = \Openkm\bean\Report::FORMAT_PDF;
        $content = $this->ws->executeReport(2, $params, $format);
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/report.pdf', 'w+');
                fwrite($file, $content);
        }
    }
}
```

```
        fclose($file);
        echo 'download correct';
        break;
    case 2:
        header('Expires', 'Sat, 6 May 1971 12:00:00 GMT');
        header('Cache-Control', 'max-age=0, must-revalidate');
        header('Cache-Control', 'post-check=0, pre-check=0');
        header('Pragma', 'no-cache');
        header('Content-Type: ' . $format);
        header('Content-Disposition: attachment; filename="out.pdf"');
        echo $content;
        break;
    }
}

}

$openkm = new OpenKM(); //autoload
$exampleReport = new ExampleReport();
$exampleReport->testExecuteReport(1);
?>
```