



Documentation for SDK for PHP 1.2.0

## Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>SDK for PHP 1.2.0</b>	<b>5</b>
License	5
Compatibility	5
Download	5
Sample client	6
Basic concepts	7
Authentication	7
Accessing API	9
Class Hierarchy	11
Auth samples	14
Basics	14
Methods	14
getGrantedRoles	14
getGrantedUsers	15
getRoles	16
getUsers	17
grantRole	18
grantUser	19
revokeRole	20
revokeUser	21
getRolesByUser	22
getUsersByRole	23
getMail	24
getName	25
Document samples	27
Basics	27
Methods	27
createDocumentSimple	27
deleteDocument	28
getDocumentProperties	29
getContent	30
getContentByVersion	31
getDocumentChildren	32
renameDocument	33
setProperties	34
createFromTemplate	35
checkout	35
cancelCheckout	36
forceCancelCheckout	37
isCheckedOut	38
checkin	39
getVersionHistory	40
lock	41
unlock	42
forceUnlock	43
isLocked	44
getLockInfo	45
purgeDocument	45
moveDocument	46
copyDocument	47
restoreVersion	48
purgeVersionHistory	49
getVersionHistorySize	50
isValidDocument	51
getDocumentPath	52
Folder samples	54
Basics	54

<b>Methods</b>	<b>54</b>
createFolder	54
createFolderSimple	55
getFolderProperties	56
deleteFolder	57
renameFolder	58
moveFolder	59
getFolderChildren	59
isValidFolder	60
getFolderPath	61
<b>Note samples</b>	<b>63</b>
Basics	63
Methods	63
addNote	63
getNote	64
deleteNote	65
setNote	66
listNotes	67
<b>Property samples</b>	<b>69</b>
Basics	69
Methods	69
addCategory	69
removeCategory	70
addKeyword	71
removeKeyword	72
setEncryption	73
unsetEncryption	74
setSigned	75
<b>PropertyGroup samples</b>	<b>77</b>
Basics	77
Methods	77
addGroup	77
removeGroup	78
getGroups	79
getAllGroups	80
getPropertyGroupProperties	81
getPropertyGroupForm	82
setPropertyGroupProperties	83
setPropertyGroupPropertiesSimple	84
hasGroup	85
<b>Repository samples</b>	<b>87</b>
Methods	87
getRootFolder	87
getTrashFolder	87
getTemplatesFolder	88
getPersonalFolder	89
getMailFolder	90
getThesaurusFolder	91
getCategoriesFolder	92
purgeTrash	93
getUpdateMessage	94
getRepositoryUuid	94
hasNode	95
getNodePath	96
getNodeUuid	97
getAppVersion	98
executeScript	99
executeSqlQuery	100
executeHqlQuery	101
<b>Search samples</b>	<b>103</b>
Basics	103
Methods	106
findByContent	106
findByName	107

---

findByKeywords	108
find	109
findPaginated	110
findSimpleQueryPaginated	111
findMoreLikeThis	113
getKeywordMap	114
getCategorizedDocuments	115
saveSearch	116
updateSearch	117
getSearch	118
getAllSearchs	119
deleteSearch	120

# SDK for PHP 1.2.0

OpenKM SDK for PHP is a set of software development tools that allows the creation of applications for OpenKM. The OpenKM SDK for PHP includes 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 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 1.2.0 should be used:

- From OpenKM Community version 6.3.7.

## Download

Download the [sdk4php-1.2.0.zip](#) file.

## 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
<b>OKMAuth</b>	Manages security and users. For example add or remove grants on a node, create or modify users or getting the profiles.	Yes	AuthImpl.php	BaseAuth.php
<b>OKMBookmark</b>	Manages the user bookmarks.	No		
<b>OKMDashboard</b>	Manages all data shown at dashboard.	No		
<b>OKMDocument</b>	Manage documents. For example creates, moves or deletes a document.	Yes	DocumentImpl.php	BaseDocument.php
<b>OKMFolder</b>	Manages folders. For example create, move	Yes	FolderImpl.php	BaseFolder.php

	or delete a folder.			
<b>OKMMail</b>	Manages mails. For example creates, moves or deletes a mails.	Yes	MailImpl.php	BaseMail.php
<b>OKMNote</b>	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
<b>OKMNotification</b>	Manages notifications. For example add or remove subscriptions on a document or a folder.	No	NotificationImpl.php	BaseNotification.php
<b>OKMProperty</b>	Manages categories and keywords. For example add or remove keywords on a document, folder, mail or record.	Yes	PropertyImpl.php	BaseProperty.php
<b>OKMPropertyGroup</b>	Manages metadata. For example add metadata group, set metadata fields.	Yes	PropertyGroupImpl.php	BasePropertyGroup.php
<b>OKMRecord</b>	Manages records. For example create, move or delete a record.	Yes	RecordImpl.php	BaseRecord.php
<b>OKMRelation</b>	Manages relations	Yes	RelationImpl.php	BaseRelation.php

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

## Class Hierarchy

Packages detail:

Name	Description
<b>com.openkm</b>	<p>The <b>openkm.OKMWebservicesFactory</b> that returns an <b>openkm.OKMWebservices</b> object which implements all interfaces.</p> <pre data-bbox="424 439 1401 495" style="border: 1px solid yellow; padding: 2px;">\$this-&gt;ws = OKMWebServicesFactory::build(host, user, password);</pre>
<b>openkm.bean</b>	Contains all classes result of unmarshalling REST objects.
<b>openkm.definition</b>	<p>All interface classes:</p> <ul style="list-style-type: none"> <li>• openkm.definition.BaseAuth</li> <li>• openkm.definition.BaseDocument</li> <li>• openkm.definition.BaseFolder</li> <li>• openkm.definition.BaseMail</li> <li>• openkm.definition.BaseNote</li> <li>• openkm.definition.BaseProperty</li> <li>• openkm.definition.BasePropertyGroup</li> <li>• openkm.definition.BaseRecord</li> <li>• openkm.definition.BaseRelation</li> <li>• openkm.definition.BaseRepository</li> <li>• openkm.definition.BaseSearch</li> <li>• openkm.definition.BaseWorkflow</li> </ul>
<b>openkm.impl</b>	<p>All interface implementation classes:</p> <ul style="list-style-type: none"> <li>• openkm.impl.AuthImpl</li> <li>• openkm.impl.DocumentImpl</li> <li>• openkm.impl.FolderImpl</li> <li>• openkm.impl.MailImpl</li> <li>• openkm.impl.NoteImpl</li> <li>• openkm.impl.PropertyGroupImpl</li> <li>• openkm.impl.PropertyImpl</li> <li>• openkm.impl.RecordImpl</li> <li>• openkm.impl.RelationImpl</li> <li>• openkm.impl.RepositoryImpl</li> <li>• openkm.impl.SearchImpl</li> </ul>

	<ul style="list-style-type: none"><li>• <code>openkm.impl.WorkflowImpl</code></li></ul>
<b>openkm.util</b>	A couple of utilities.
<b>openkm.exception</b>	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

#### getGrantedRoles

Description:

Method	Return values	Description
<b>getGrantedRoles(\$nodeId)</b>	<b>array</b>	Returns the granted roles of a node.
<b>Parameters:</b>		
<b>\$nodeId</b> 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
<b>getGrantedUsers(\$nodeId)</b>	<b>array</b>	Returns the granted users of a node.
<b>Parameters:</b>		
<b>\$nodeId</b> 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
<b>getRoles()</b>	<b>array</b>	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
<b>getUsers()</b>	<b>array</b>	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
<b>grantRole(\$nodeId, \$role, \$permissions, \$recursive)</b>	<b>void</b>	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 the 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::WRITE);
        } catch (Exception $e) {
            // ...
        }
    }
}
```

```

        // Add all ROLE_ADMIN grants to the node and descendants
        $this->ws->grantRole("/okm:root/SDK4PHP", "ROLE_ADMIN", openkm\bean\Permi.

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

}

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

```

## grantUser

Description:

Method	Return values	Description
<b>grantUser(\$nodeId, \$user, \$permissions, \$recursive)</b>	<b>void</b>	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 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 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
<b>revokeRole(\$nodeId, \$role, \$permissions, \$recursive)</b>	<b>void</b>	Removes a role grant on a node.
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record.</p> <p><b>\$role</b> string type</p> <p><b>\$permissions</b> int type</p> <p><b>\$recursive</b> bool type</p> <p>The parameter recursive only has sense when the nodeId is a folder or record node.</p> <p>When the 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 testRevokeRole() {
        try {
            // Remove ROLE_USER write grants at the node but not descendants
            $this->ws->revokeRole("/okm:root/SDK4PHP", "ROLE_USER", \openkm\bean\Permissions.class);

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

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

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

```

## revokeUser

Description:

Method	Return values	Description
<b>revokeUser(\$nodeId, \$user, \$permissions, \$recursive)</b>	<b>void</b>	Removes a user grant on a node.
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record.</p> <p><b>\$user</b> string type</p> <p><b>\$permissions</b> int type</p> <p><b>\$recursive</b> bool type</p> <p>The parameter recursive only has 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
<b>getRolesByUser(\$user)</b>	<b>array</b>	Returns the list of all the roles assigned to a user.
<b>Parameters:</b>		
<b>\$user</b> 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
<b>getUsersByRole(\$role)</b>	<b>array</b>	Returns the list of all the users who have assigned a role.
<b>Parameters:</b>		
<b>\$role</b> 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
<b>getMail(\$user)</b>	<b>string</b>	Returns the mail of a valid user.
<b>Parameters:</b>		
<b>\$user</b> 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
<b>getName(\$user)</b>	<b>string</b>	Returns the name of a valid user.
<b>Parameters:</b>		
<b>\$user</b> 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();
?>
```

## 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
<b>createDocumentSimple(\$docPath, \$content)</b>	<b>Document</b>	Creates a new document and return as a result an object Document with the properties of the created document.
<p><b>Parameters:</b></p> <p><b>\$docPath</b> string type is the Path of the Document</p> <p><b>\$content</b> string type is recommend using <code>file_get_contents</code> — 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
<b>deleteDocument(\$docId)</b>	<b>void</b>	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
<b>getDocumentProperties(\$docId)</b>	<b>Document</b>	Returns the document properties.
<b>Parameters:</b>		
<b>\$docId</b> 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
<b>getContent(\$docId)</b>	<b>string</b>	Retrieves a document content - binary data - of the actual document version.
<b>Parameters:</b>		
<b>\$docId</b> 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
<b>getContentByVersion(\$docId, \$versionId)</b>	<b>string</b>	Retrieves a document content ( binary data ) of an specific document version.
<p><b>Parameters:</b></p> <p><b>\$docId</b> string type is the uuid or path of the Document</p> <p><b>\$versionId</b> 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);
?>

```

## getDocumentChildren

Description:

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

    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
<b>renameDocument(\$docId, \$newName)</b>	<b>Document</b>	Changes the name of a document and returns the Document
<p><b>Parameters:</b></p> <p><b>\$docId</b> string type is the uuid or path of the Document</p> <p><b>\$newName</b> string type is the new name for 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 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
<b>setProperties(Document \$doc)</b>	<b>void</b>	Changes a document properties.

Variables allowed to be changed:

- 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](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::PASSWO
}

public function testSetProperties(){
    try {
        $document = $this->ws->getDocumentProperties('604e17e8-3285-4e8a-910c-c99
        $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();
?>

```

**createFromTemplate**

**checkout**

Description:

Method	Return values	Description
<b>checkout(\$docId)</b>	<b>void</b>	Marks the document for edition.

**Parameters:**

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

Only one user can modify a document at same time.

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

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 checked out
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

```


**cancelCheckout**

Description:

Method	Return values	Description
<b>cancelCheckout(\$docId)</b>	<b>void</b>	Cancels a document edition.

**Parameters:**

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

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

Example:

```

<?php

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

use openkm\OKMWebServicesFactory;

```

```

use openkm\OpenKM;

class 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
            $this->ws->cancelCheckout('/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->testCancelCheckout();
?>

```

### forceCancelCheckout

Description:


Method	Return values	Description
<b>forceCancelCheckout(\$docId)</b>	<b>void</b>	Cancel a document edition.

**Parameters:**

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

This method allows to cancel the edition of any document.

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

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

Example:

```
<?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
<b>isCheckedOut(\$docId)</b>	<b>bool</b>	Returns a boolean that indicate if the document is on edition or not.
<b>Parameters:</b>		
<b>\$docId</b> 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 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
<b>checkin(\$docId, \$content, \$comment)</b>	<b>Version</b>	Updates a document with a new version and returns an object with new Version values.

**Parameters:**

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

**\$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

 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
<b>getVersionHistory(\$docId)</b>	<b>array</b>	Returns a list of all document versions.
<b>Parameters:</b>		
<b>\$docId</b> 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
<b>lock(\$docId)</b>	<b>LockInfo</b>	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
<b>unlock(\$docId)</b>	<b>void</b>	Unlocks a locked document.

**Parameters:**

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

 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
<b>forceUnlock(\$docId)</b>	<b>void</b>	Unlocks a locked document.

**Parameters:**

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

This method allows to unlcok any locked document.

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

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

Example:

```

<?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 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
<b>isLocked(\$docId)</b>	<b>bool</b>	Returns a boolean that indicates if the document is locked or not.
<b>Parameters:</b>		
<b>\$docId</b> 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/logo.
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

```

```

}

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

```

### getLockInfo

Description:

Method	Return values	Description
<b>getLockInfo(\$docId)</b>	<b>LockInfo</b>	Return an object with the Lock information
<b>Parameters:</b>		
<b>\$docId</b> 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
<b>purgeDocument(\$docId)</b>	<b>void</b>	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 is possible to directly purge any document from the whole repository.

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

Example:

```
<?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
<b>moveDocument(\$docId, \$dstId)</b>	<b>void</b>	Moves a document into some 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

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/');
            echo 'moveDocument';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

## copyDocument

Description:

Method	Return values	Description
<b>copyDocument(\$docId, \$dstId)</b>	<b>void</b>	Copies a document into a folder or record.

<p><b>Parameters:</b></p> <p><b>\$docId</b> string type is the uuid or path of the Document</p> <p><b>\$dstId</b> 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 testCopyDocument() {
        try {
            $this->ws->copyDocument('/okm:root/SDK4PHP/logo.png', '/okm:root/SDK4PHP/');
            echo 'copyDocument';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

**restoreVersion**

Description:

Method	Return values	Description
<b>restoreVersion(\$docId, \$versionId)</b>	<b>void</b>	Promotes previous document version to actual version.
<p><b>Parameters:</b></p> <p><b>\$docId</b> string type is the uuid or path of the Document</p>		



**\$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::PASSWORD);
    }

    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
<b>purgeVersionHistory(\$docId)</b>	<b>void</b>	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
<b>getVersionHistorySize(\$docId)</b>	<b>int</b>	Returns the sum in bytes of all documents into documents history.
<p><b>Parameters:</b></p> <p><b>\$docId</b> 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 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
<b>isValidDocument(\$docId)</b>	<b>bool</b>	Returns a boolean that indicates if the node is a document or not.
<b>Parameters:</b>		
<b>\$docId</b> 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($this->ws->isValidDocument("/okm:root/SDK4PHP/logo.png"));
            // Return false
            var_dump($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
<b>getDocumentPath(\$uuid)</b>	<b>string</b>	Convert document UUID to document path.
<b>Parameters:</b>		
<b>\$uuid</b> 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();
?>
```

## Folder samples

### Basics

On most 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
<b>createFolder(Folder \$fld)</b>	<b>Folder</b>	Creates a new folder and return as 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 the folder creation.

We suggest using the method below createFolderSimple 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
<b>createFolderSimple(\$fldPath)</b>	<b>Folder</b>	Creates a new folder and returns as a result an object Folder.
<b>Parameters:</b>		
<b>\$fldPath</b> 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
<b>getFolderProperties(\$fldId)</b>	<b>Folder</b>	Returns the folder properties.
<b>Parameters:</b>		
<b>\$fldId</b> 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
<b>deleteFolder(\$fldId)</b>	<b>void</b>	Deletes a folder.
<p><b>Parameters:</b></p> <p><b>\$fldId</b> 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
<b>renameFolder(\$fldId, \$newName)</b>	<b>void</b>	Renames a folder.
<p><b>Parameters:</b></p> <p><b>\$fldId</b> string type is the uuid or path of the Folder</p> <p><b>\$newName</b> 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
<b>moveFolder(\$fldId, \$dstId)</b>	<b>void</b>	Moves a folder into a folder or record.
<p><b>Parameters:</b></p> <p><b>\$fldId</b> string type is the uuid or path of the Folder</p> <p><b>\$dstId</b> string type is the uuid or path of the Folder or record</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 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
<b>getFolderChildren(\$fldId)</b>	<b>array</b>	Returns an array of all folder which their parent is fldId.
<p><b>Parameters:</b></p> <p><b>\$fldId</b> 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

<b>isValidFolder(\$fId)</b>	<b>Boolean</b>	Returns a boolean that indicates if the node is a folder or not.
<b>Parameters:</b>		
<b>\$fId</b> 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
<b>getFolderPath(\$uuid)</b>	<b>String</b>	Converts folder UUID to folder path.
<b>Parameters:</b>		

**\$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();

?>
```

## 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
<b>addNote(\$nodeId, \$text)</b>	<b>Note</b>	Adds a note to a node and returns an object Note.
<p><b>Parameters:</b></p> <p><b>\$fldPath</b> string type is the uuid or path of the document, folder, mail or record</p> <p><b>\$text</b> 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
<b>getNode(\$noteId)</b>	<b>Note</b>	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
<b>deleteNote(\$noteId)</b>	<b>Note</b>	Deletes a note.

**Parameters:**

**\$noteId** string type

**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::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
<b>setNote(\$noteId, \$text)</b>	<b>void</b>	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
<b>listNotes(\$nodeId)</b>	<b>array</b>	Retrieves a list of all notes of a node.
<b>Parameters:</b>		
<b>\$nodeId</b> 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
<b>addCategory(\$nodeId, \$catId)</b>	<b>void</b>	Sets a relation between a category and a node.
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record</p> <p><b>\$catId</b> 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
<b>removeCategory(\$nodeId, \$catId)</b>	<b>void</b>	Removes a relation between a category and a node.
<b>Parameters:</b>		
<b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record		
<b>\$catId</b> 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
<b>addKeyword(\$nodeId, \$keyword)</b>	<b>void</b>	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
<b>removeKeyword(\$nodeId, \$keyword)</b>	<b>void</b>	Removes a keyword from a node.
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record</p> <p><b>\$keyword</b> 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
<b>setEncryption(\$nodeId, \$cipherName)</b>	<b>void</b>	Marks a document as an encrypted binary data into the repository

**Parameters:**

**\$nodeId** string type is the uuid or path of the document

**\$cipherName** string type is the cipher name saves information about the encryption mechanism.

 This method does not perform any kind of encryption, it simply marks into the database that a document is encrypted.

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
<b>unsetEncryption(\$nodeId)</b>	<b>void</b>	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, it simply marks 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
<b>setSigned(\$nodeId, \$signed)</b>	<b>void</b>	Marks a document as signed or unsigned binary data into the repository
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document</p> <p><b>\$signed</b> 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, it simply marks 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 does not help 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
<b>addGroup(\$nodeId, \$grpName)</b>	<b>void</b>	Add an empty metadata group to a node.
<b>Parameters:</b>		
<b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record.		
<b>\$grpName</b> 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
<b>removeGroup(\$nodeId, \$grpName)</b>	<b>void</b>	Removes a metadata group of 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 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
<b>getGroups(\$nodeId)</b>	<b>array</b>	Retrieves a list of metadata groups assigned to a node.
<b>Parameters:</b>		
<b>\$nodeId</b> 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
<b>getAllGroups()</b>	<b>array</b>	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
<b>getPropertyGroupProperties(String nodeId, String grpName)</b>	<b>List&lt;FormElement&gt;</b>	Retrieves a list of all metadata group elements and its values of a node.
<p><b>Parameters:</b></p> <p><b>\$nodeId</b> string type is the uuid or path of the document, folder, mail or record.</p> <p><b>\$grpName</b> 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 used.         </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();
?>
```


## getPropertyGroupForm

Description:

Method	Return values	Description
<b>getPropertyGroupForm(\$grpName)</b>	<b>array</b>	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
<b>setPropertyGroupProperties(\$nodeId, \$grpName, \$formElements)</b>	<b>void</b>	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



Is not mandatory set into parameter of `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', 'okg:
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

$openkm = new OpenKM(); //autoload
$examplePropertyGroup = new ExamplePropertyGroup();
$examplePropertyGroup->testSetPropertyGroupProperties();
?>

```

### setPropertyGroupPropertiesSimple

Description:

Method	Return values	Description
<b>setPropertyGroupPropertiesSimple(\$nodeId, \$grpName, \$properties)</b>	<b>void</b>	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 of the SimplePropertyGroup

**i** Is not mandatory set into properties parameter all fields values, is enough with the fields you wish to change its value

**i** 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
<b>hasGroup(\$nodeId, \$grpName)</b>	<b>bool</b>	Returns a boolean that indicate if the node has or not a 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.

## 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();
?>
```

## Repository samples

### Methods

#### getRootFolder

Description:

Method	Return values	Description
<b>getRootFolder()</b>	<b>Folder</b>	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

<b>getTrashFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:trash/{userId}"
<p>The returned folder will be the user trash folder.</p> <div style="border: 1px dashed #00aaff; padding: 10px; background-color: #e6f2ff;"> <p><b>i</b> For example if the method is executed by "okmAdmin" user then the folder returned will be "/okm:trash/okmAdmin".</p> </div>		

**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();
?>
```

**getTemplatesFolder**

**Description:**

Method	Return values	Description
<b>getTemplatesFolder()</b>	<b>Folder</b>	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
<b>getPersonalFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.



For example if the method is executed by "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();
?>

```

**getMailFolder**

Description:

Method	Return values	Description
<b>getMailFolder()</b>	<b>Folder</b>	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

**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();
?>

```

### getThesaurusFolder

Description:

Method	Return values	Description
<b>getThesaurusFolder()</b>	<b>Folder</b>	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
<b>getCategoriesFolder()</b>	<b>Folder</b>	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
<b>purgeTrash()</b>	<b>void</b>	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 it will only be able to be restored from a previously repository backup. The purge action removes 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
<b>getUpdateMessage()</b>	<b>string</b>	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
<b>getRepositoryUuid()</b>	<b>string</b>	Retrieves an 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
<b>hasNode(\$nodeId)</b>	<b>bool</b>	Returns a node that indicates if a node exists or not.
<b>Parameters:</b>		
<b>\$nodeId</b> 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
<b>getNodePath(\$uuid)</b>	<b>string</b>	Converts a node UUID to path.
<b>Parameters:</b>		
<b>\$uuid</b> 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
<b>getNodeUuid(\$nodePath)</b>	<b>string</b>	Converts a node path to UUID.
<b>Parameters:</b>		
<b>\$nodePath</b> 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
<b>getAppVersion()</b>	<b>AppVersion</b>	Returns information about an 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();
?>

```

## executeScript

Description:

Method	Return values	Description
<b>executeScript(\$content)</b>	<b>ScriptExecutionResult</b>	Executes an script.


**Parameters:**

**\$content** string type Recommend 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 = $this->ws->executeScript(file_get_contents($file));
        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
<b>executeSqlQuery(\$content)</b>	<b>SqlQueryResults</b>	Executes SQL sentences.

**Parameters:**

**\$content** string type Recommends using file\_get\_contents — Reads entire file into a string

The test.sql script used in the sample below:

```
SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;
```

 The SQL script can only contains a single SQL sentence.  
 This action can only be done by a super user ( user with ROLE\_ADMIN ).

Example:

```

<?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
<b>executeHqlQuery(\$content)</b>	<b>HqlQueryResults</b>	Executes HQL sentences.
<p><b>Parameters:</b></p> <p><b>\$content</b> string type Recommends using <code>file_get_contents</code> — Reads entire file into a string</p> <p>The <code>testhql.sql</code> script used in the sample below:</p> <pre>SELECT uuid, name from NodeBase where name = 'okm:root';</pre> <p style="background-color: #fff9c4; padding: 5px; border: 1px dashed #ccc; text-align: center;">The HQL script can only contains a single HQL sentence.</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\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);
```

## Search samples

### Basics


Most methods use QueryParams here there're some tips about how using it.

Variables	Type	Allow wildcards	Restrictions
<b>domain</b>	<b>int</b>	No.	<p>Available values:</p> <ul style="list-style-type: none"> <li>• QueryParams::DOCUMENT</li> <li>• QueryParams::FOLDER</li> <li>• QueryParams::MAIL</li> <li>• QueryParams::RECORD</li> </ul> <p>By default the value is set to QueryParams.DOCUMENT.</p> <p>For searching documents and folders use value:</p> <pre>(QueryParams::DOCUMENT   QueryParams::FOLDER)</pre>
<b>author</b>	<b>string</b>	No.	Value must be a valid userId.
<b>name</b>	<b>string</b>	Yes.	
<b>title</b>	<b>string</b>	Yes.	
<b>keywords</b>	<b>array</b>	Yes.	
<b>categories</b>	<b>array</b>	No.	Values should be categories UUID, not use path value.

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



<b>mailFrom</b>	<b>string</b>	Yes.	Only apply to mail nodes.
<b>mailTo</b>		Yes.	Only apply to mail nodes.
<b>notes</b>		Yes.	
<b>properties</b>	<b>array</b>	Yes on almost.	<p>On metadata field values like "date" can not be applied wilcards.</p> <p>The map of the properties is composed of pairs: (<code>metadata_field_name</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
<b>name</b>	<b>test*.html</b>	Any document that start with characters "test" and ends with characters ".html"
<b>name</b>	<b>test?.html</b>	Any document that start with characters "test" followed by a single character and ends with characters ".html"
<b>name</b>	<b>?test*</b>	Any of the documents where the first character doesn't matter, but is followed by the characters, "test".

## Methods


### findByContent

Description:

Method	Return values	Description
<b>findByContent(\$content)</b>	<b>array</b>	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 does 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
<b>findByName(\$name)</b>	<b>array</b>	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
<b>findByKeywords(\$keywords)</b>	<b>array</b>	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 does 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
<b>find(QueryParams \$queryParams)</b>	<b>array</b>	Returns a list of QueryResults filtered by the values of the queryParams parameter.
<p><b>Parameters:</b></p> <p><b>\$queryParams</b> QueryParams type</p>		

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);
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}
```



```

$openkm = new OpenKM(); //autoload
$exampleSearch = new ExampleSearch();
$exampleSearch->testFind();
?>

```

## findPaginated

Description:

Method	Return values	Description
<b>findPaginated(QueryParams \$queryParams, \$offset, \$limit)</b>	<b>ResultSet</b>	Returns a list of paginated results filtered by the values of the queryParams parameter.
<p><b>Parameters:</b></p> <p><b>\$queryParams</b> QueryParams type</p> <p><b>\$offset</b> int type</p> <p><b>\$limit</b> int type</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"> <li>• The parameter "limit" is used to limit the number of results returned.</li> <li>• The parameter "offset" says to skip that many results before the beginning to return results.</li> </ul> </div> <div style="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"> <li>• limit=10</li> <li>• offset=0</li> </ul> <p>Now suppose you want to show the results from 11-20, you should use these values:</p> <ul style="list-style-type: none"> <li>• limit=10</li> <li>• offset=10</li> </ul> </div>		

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);
            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();
?>

```

### findSimpleQueryPaginated

Description:

Method	Return values	Description
<b>findSimpleQueryPaginated(\$statement, \$offset, \$limit)</b>	<b>ResultSet</b>	Returns a list of paginated results filtered by the values of the statement parameter.
<p><b>Parameters:</b></p> <p><b>\$statement</b> string type</p> <p><b>\$offset</b> int type</p> <p><b>\$limit</b> int type</p>		



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
$exampleSearch = new ExampleSearch();
$exampleSearch->testFindSimpleQueryPaginated();
?>

```

**findMoreLikeThis**

Description:

Method	Return values	Description
<b>findMoreLikeThis(String uuid, int max)</b>	<b>ResultSet</b>	Returns a list of documents that are considered similar by search engine.
<p><b>Parameters:</b></p> <p><b>\$statement</b> string type is the uuid of the Document</p> <p><b>\$offset</b> int type is the max value is used to limit the number of results returned.</p> <div style="border: 1px dashed orange; background-color: #fff9c4; padding: 5px; margin-top: 10px;">  <b>The method can only be used with documents.</b> </div>		

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");
        } catch (Exception $e) {
            // Handle exception
        }
    }
}

```

```

        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
<b>getKeywordMap(\$filter)</b>	<b>array</b>	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
<b>getCategorizedDocuments(String categoryId)</b>	<b>List&lt;Document&gt;</b>	Retrieves a list of all documents related with a category.
<b>Parameters:</b>		
<b>\$categoryId</b> 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
<b>saveSearch(QueryParams \$params)</b>	<b>int</b>	Saves a search parameters and returns the id of the saved search
<b>Parameters:</b>		
<b>\$params</b> 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);
            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
<b>updateSearch(QueryParams \$params)</b>	<b>void</b>	Updates a previously saved search parameters.
<b>Parameters:</b>		
<b>\$params</b> QueryParams type		
<div style="border: 1px dashed orange; padding: 5px; background-color: #fff9c4;">  Only can be updated a saved search created by the same user user who's executing the method.                 </div>		

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
<b>getSearch(\$qpId)</b>	<b>QueryParams</b>	Gets a saved search parameters.
<b>Parameters:</b>		
<b>\$qpId</b> 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
<b>getAllSearchs()</b>	<b>List&lt;QueryParams&gt;</b>	Retrieve 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
<b>deleteSearch(int qpId)</b>	<b>void</b>	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();
?>
```