Documentation for SDK for PHP 1.1.1

Table of Contents

SDK for PHP 1.1.1

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 1.1.0 should be used:

- **From OpenKM Professional version 6.2.19 to 6.2.28.**
- **From OpenKM Community version 6.3.0 and upper.**

Download

Download the sdk4php-1.1.1.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::
    }

    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::
    }

    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::US
    }

    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	Int
OKMAuth	Manages security and users. For example add or remove grants on a node, create or modify	Yes	AuthImpl.php	BaseAuth.

	users or getting the profiles.			
OKMBookmark	Manages the user bookmarks.	No		
OKMDashboard	Manages all data shown at dashboard.	No		
OKMDocument	Manage documents. For example creates, moves or deletes a document.	Yes	DocumentImpl.php	BaseDocu
OKMFolder	Manages folders. For example create, move or delete a folder.	Yes	FolderImpl.php	BaseFolde
OKMMail	Manages mails. For example creates,	Yes	MailImpl.php	BaseMail.

	moves or deletes a mails.			
OKMNote	Manages notes on any node type. For example creates, edits or deletes a note on a document, folder, mail or record.	Yes	NoteImpl.php	BaseNote.
OKMNotification	Manages notifications. For example add or remove subscriptions on a document or a folder.	No	NotificationImpl.php	BaseNotifi
OKMProperty	Manages categories and keywords. For example add or	Yes	PropertyImpl.php	BasePrope

	remove keywords on a document, folder, mail or record.			
OKMPropertyGroup	Manages metadata. For example add metadata group, set metadata fields.	Yes	PropertyGroupImpl.php	BasePrope
OKMRecord	Manages records. For example create, move or delete a record.	Yes	RecordImpl.php	BaseRecor
OKMRelation	Manages relations between nodes. For example create a relation (parent-child) between two documents.	Yes	RelationImpl.php	BaseRelati

OKMRepository	A lot of stuff related with repository. For example get the properties of main root node (/okm:root).	Yes	RepositoryImpl.php	BaseRepos
OKMSearch	Manages search feature. For example manage saved queries or perform a new query to the repository.	Yes	SearchImpl.php	BaseSearc
OKMStats	General stats of the repository.	No		
OKMTask	Manages task. For example create a new task.	No		
OKMUserConfig	Manages user	No		

	home configuration.			
OKMWorkflow	Manages workflows. For example executes a new workflow.	Yes	WorkflowImpl.php	BaseWork

Class Hierarchy

Packages detail:

Name	Description
com.openkm	The **openkm.OKMWebservicesFactory** that returns an **openkm.OKMWebservices** object which implements all interfaces. `$this->ws = OKMWebServicesFactory::build(host, user, password);`
openkm.bean	Contains all classes result of unmarshalling REST objects.
openkm.definition	All interface classes: • openkm.definition.BaseAuth • openkm.definition.BaseDocument • openkm.definition.BaseFolder • openkm.definition.BaseMail • openkm.definition.BaseNote • openkm.definition.BaseProperty • openkm.definition.BasePropertyGroup

	• openkm.definition.BaseRecord • openkm.definition.BaseRelation • openkm.definition.BaseRepository • openkm.definition.BaseSearch • openkm.definition.BaseWorkflow
openkm.impl	All interface implementation classes: • openkm.impl.AuthImpl • openkm.impl.DocumentImpl • openkm.impl.FolderImpl • openkm.impl.MailImpl • openkm.impl.NoteImpl • openkm.impl.PropertyGroupImpl • openkm.impl.PropertyImpl • openkm.impl.RecordImpl • openkm.impl.RelationImpl • openkm.impl.RepositoryImpl • openkm.impl.SearchImpl • openkm.impl.WorkflowImpl
openkm.util	A couple of utilities.
openkm.exception	All exception classes.

Auth samples

Basics

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

To set READ and WRITE access you should do:

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

To check if you have permission access you should do:

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

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

Example of nodeId:

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

Methods

getGrantedRoles

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

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

}

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

getGrantedUsers

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

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

}

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

getRoles

Description:

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

Example:

```
<?php

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

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

class ExampleAuth {

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

    private $ws;

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

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

}

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

getUsers

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

```
    const PASSWORD = "admin";

    private $ws;

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

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

}

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

grantRole

Description:

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

Parameters:

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

$role string type

$permissions int type

$recursive bool type

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

When 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::
    }

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

            // Add all ROLE_ADMIN grants to the node and descendant
            $this->ws->grantRole("/okm:root/SDK4PHP", "ROLE_ADMIN",

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

}

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

grantUser

Description:

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

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

$user string type

$permissions int type

$recursive bool type

The parameter recursive only 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::
    }

    public function testGrantUser() {
        try {
            // Add sochoa write grants at the node but not descenda
            $this->ws->grantUser("/okm:root/SDK4PHP", "sochoa", \op

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

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

}

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

revokeRole

Description:

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

Parameters:

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

$role string type

$permissions int type

$recursive bool type

The parameter recursive only 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::
    }

    public function testRevokeRole() {
        try {
```

```
            // Remove ROLE_USER write grants at the node but not de
            $this->ws->revokeRole("/okm:root/SDK4PHP", "ROLE_USER",

            // Remove all ROLE_ADMIN grants to the node and descend
            $this->ws->revokeRole("/okm:root/SDK4PHP", "ROLE_ADMIN"

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

}

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

revokeUser

Description:

Method	Return values	Description
revokeUser($nodeId, $user, $permissions, $recursive)	**void**	Removes a user grant on a node.

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::
    }

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

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

}

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

getRolesByUser

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

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

}

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

getUsersByRole

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

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

}

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

getMail

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

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

}

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

getName

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleAuth {

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

    private $ws;

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

    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

createDocument

Description:

Method	Return values	Description
createDocument(Document $document, $content)	**Document**	Creates a new document and returns as a result an object Document with the properties of the created document.

Parameters:

$document Document type is an Object Document

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

The variable **path** into the parameter **doc**, must be initializated. It indicates where the document must be stored into OpenKM.

```
$doc = new Document();
$doc->setPath("/okm:root/logo.png");
```

The other variables of a Document (doc) will not take any effect on document creation.

We suggest using the method below to createDocumentSimple rather this one.

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::
    }

    public function testCreateDocument() {
        try {
            $fileName = dirname(__FILE__) . '/files/logo.png';
            $document = new \openkm\bean\Document();
            $document->setPath('/okm:root/SDK4PHP/logo.png');
            $doc = $this->ws->createDocument($document, file_get_cc
            var_dump($doc);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

createDocumentSimple

Description:

Method	Return values	Description
createDocumentSimple($docPath, $content)	**Document**	Creates a new document and return as a result an object Document with the properties of the created document.

Parameters:

$docPath string type is the Path of the Document

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class 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::
    }

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

}

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

deleteDocument

Description:

Method	Return values	Description

deleteDocument($docId)	**void**	Deletes a document.

Parameters:

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

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

getDocumentProperties

Description:

Method	Return values	Description

getDocumentProperties($docId)	**Document**	Returns the document properties.

Parameters:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

getContent

Description:

Method	Return values	Description
		Retrieves a document content - binary data - of the

getContent($docId)	**string**	actual document version.

Parameters:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

    public function testGetContent($method) {
        $content = $this->ws->getContent('/okm:root/SDK4PHP/logo.pn
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_down
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                $document = $this->ws->getDocumentProperties('/okm:
                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="
                echo $content;
                break;
        }
    }

}

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

getContentByVersion

Description:

Method	Return values	Description
getContentByVersion($docId, $versionId)	**string**	Retrieves a document content (binary data) of an specific document version.

Parameters:

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

$versionId 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::
    }

    public function testGetContentByVersion($method) {
        $content = $this->ws->getContentByVersion('/okm:root/SDK4PH
        switch ($method) {
            case 1:
                $file = fopen(dirname(__FILE__) . '/files/logo_down
                fwrite($file, $content);
                fclose($file);
                echo 'download correct';
                break;
            case 2:
                $document = $this->ws->getDocumentProperties('/okm:
                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="
                echo $content;
                break;
            }
        }

}

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

getDocumentChildren

Description:

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

    public function testGetDocumentChildren(){
        try {
```

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

}

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

renameDocument

Description:

Method	Return values	Description
renameDocument($docId, $newName)	**Document**	Changes the name of a document and returns the Document

Parameters:

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

$newName string type is the new name for the Document

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

```
    }

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

}

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

setProperties

Description:

Method	Return values	Description
setProperties(Document $doc)	**void**	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.

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::
    }

    public function testSetProperties(){
        try {
            $document = $this->ws->getDocumentProperties('604e17e8-
            $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:categc
            $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

checkout($docId)	void	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::
    }

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

}

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

cancelCheckout

Description:

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

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::
    }

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

}

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

forceCancelCheckout

Description:

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

Parameters:

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

This method allows to cancel 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::
    }

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

```
}

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

isCheckedOut

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

    public function testIsCheckedOut() {
        try {
            echo "Is the document checkout:" . $this->ws->isChecked
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}
```

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

checkin

Description:

Method	Return values	Description
checkin($docId, $content, $comment)	**Version**	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::
    }
```

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

}

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

getVersionHistory

Description:

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

Example:

```
<?php

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

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

lock

Description:

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

Parameters:

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

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

A locked document can not be modified by other users.

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

```
    const PASSWORD = "admin";

    private $ws;

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

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

}

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

unlock

Description:

Method	Return values	Description
unlock($docId)	**void**	Unlocks a locked document.

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::
    }

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

}

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

forceUnlock

Description:

Method	Return values	Description
forceUnlock($docId)	**void**	Unlocks a locked document.

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::
    }

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

}

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

isLocked

Description:

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

Example:

```
<?php
```

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

getLockInfo

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
```

```
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

purgeDocument

Description:

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

Parameters:

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

Usually you will purge documents into /okm:trash/userId - the personal trash user locations - but 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::
    }

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

}

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

moveDocument

Description:

Method	Return values	Description
moveDocument($docId, $dstId)	**void**	Moves a document into some folder or record.
Parameters: **$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::
    }

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

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

copyDocument

Description:

Method	Return values	Description
copyDocument($docId, $dstId)	**void**	Copies a document into a folder or record.
Parameters: **$docId** string type is the uuid or path of the Document		

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

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::
    }

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

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

restoreVersion

Description:

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

$versionId string type is the version of the Document

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

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

purgeVersionHistory

Description:

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

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

This action compact the version history of a document.

This action can not be reverted.

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

getVersionHistorySize

Description:

Method	Return values	Description

getVersionHistorySize($docId)	**int**	Returns the sum in bytes of all documents into documents history.

Parameters:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

            $bytes = $this->ws->getVersionHistorySize('/okm:root/$D
            $value = "";

            for ($i = 6; $i > 0; $i--) {
                $step = pow(1024, $i);
                if ($bytes > $step) {
                    $value = number_format($bytes / $step, 2) . ' '
                }
                if (empty($value)) {
                    $value = $bytes . ' ' . $units[0];
                }
            }
            echo $value;
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

```
?>
```

isValidDocument

Description:

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

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

}

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

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

getDocumentPath

Description:

Method	Return values	Description
getDocumentPath($uuid)	**string**	Convert document UUID to document path.

Parameters:

$uuid string type is the uuid of the Document

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleDocument {

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

    private $ws;

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

    public function testGetDocumentPath() {
        try {
            var_dump($this->ws->getDocumentPath("8b559709-3c90-4c26
        } 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
createFolder(Folder $fld)	**Folder**	Creates a new folder and return as a result an object Folder.

The variable **path** into the parameter **$fld**, must be initializated. 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::
    }

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

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

createFolderSimple

Description:

Method	Return values	Description
createFolderSimple($fldPath)	**Folder**	Creates a new folder and returns as a result an object Folder.
Parameters: **$fldPath** string type is the Path of the Folder		

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

    public function testCreateFolderSimple() {
        try {
            $folder = $this->ws->createFolderSimple("/okm:root/SDK4
            var_dump($folder);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

?>
```

getFolderProperties

Description:

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

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

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

}

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

?>
```

deleteFolder

Description:

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

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

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

}

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

?>
```

renameFolder

Description:

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

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

}

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

moveFolder

Description:

Method	Return values	Description
moveFolder($fldId, $dstId)	**void**	Moves a folder into a folder or record.

Parameters:

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

$dstId string type is the uuid or path of the Folder or record

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

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

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

?>
```

getFolderChildren

Description:

Method	Return values	Description
getFolderChildren($fldId)	**array**	Returns an array of all folder which their parent is fldId.
Parameters:		

$fldId string type is the uuid or path of the Folder or Record node

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::
    }

    public function testGetFolderChildren(){
        try {
            $folders = $this->ws->getFolderChildren("/okm:root/SDK4
            foreach ($folders as $folder) {
                var_dump($folder);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

?>
```

isValidFolder

Description:

Method	Return values	Description
isValidFolder($fldId)	**Boolean**	Returns a boolean that indicates if the node is a folder or not.

Parameters:

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

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

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

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

?>
```

getFolderPath

Description:

Method	Return values	Description
getFolderPath($uuid)	**String**	Converts folder UUID to folder path.

Parameters:

$uuid string type is the uuid of the Folder

Example:

```
<?php

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

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

class ExampleFolder {

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

    private $ws;

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

    public function testGetFolderPath(){
        try {
            var_dump($this->ws->getFolderPath("6e86cc0b-bc9a-4c51-8
        } 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
addNote($nodeId, $text)	**Note**	Adds a note to a node and returns an object Note.
Parameters: **$fldPath** string type is the uuid or path of the document, folder, mail or record **$text** string type is the text		

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::
    }

    public function testAddNote() {
        try {
            $note = $this->ws->addNote("/okm:root/SDK4PHP/logo.png"
            var_dump($note);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testAddNote();

?>
```

getNote

Description:

Method	Return values	Description
getNote($noteId)	**Note**	Retrieves the note.

Parameters:

$noteId string type

> 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::
    }

    public function testGetNote() {
        try {
            $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.p
            if (count($notes) > 0) {
                var_dump($this->ws->getNote($notes[0]->getPath()));
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testGetNote();

?>
```

deleteNote

Description:

Method	Return values	Description
deleteNote($noteId)	**Note**	Deletes a note.

Parameters:

$noteId string type

> 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::
    }

    public function testDeleteNote(){
        try {
            $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.p
            if (count($notes) > 0) {
                $this->ws->deleteNote($notes[0]->getPath());
                echo "deleted";
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testDeleteNote();

?>
```

setNote

Description:

Method	Return values	Description
setNote($noteId, $text)	**void**	Changes the note text.
Parameters:		

$noteId string type

$text string type is the text

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::
    }

    public function testSetNote(){
        try {
            $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.p
            if (count($notes) > 0) {
                $this->ws->setNote($notes[0]->getPath(),"text modif
                echo "updated";
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testSetNote();

?>
```

listNotes

Description:

Method	Return values	Description
listNotes($nodeId)	**array**	Retrieves a list of all notes of a node.
Parameters: **$nodeId** string type is the uuid or path of the document, folder, mail or record.		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\Note;

class ExampleNote {

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

    private $ws;

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

    public function testListNotes() {
        try {
            $notes = $this->ws->listNotes("/okm:root/SDK4PHP/logo.p
            foreach ($notes as $note) {
                var_dump($note);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleNote = new ExampleNote();
$exampleNote->testListNotes();
?>
```

Property samples

Basics

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

Example of nodeId:

- Using UUID -> "**adabdb0f-7ff8-4832-9e43-8bc96fc1c9a5**";
- Using path -> "**/okm:root/SDK4PHP/logo.png**"

Methods

addCategory

Description:

Method	Return values	Description
addCategory($nodeId, $catId)	**void**	Sets a relation between a category and a node.
Parameters: **$nodeId** string type is the uuid or path of the document, folder, mail or record **$catId** string type is the text		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {
```

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

    private $ws;

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

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

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveCategory();

?>
```

removeCategory

Description:

Method	Return values	Description
removeCategory($nodeId, $catId)	**void**	Removes a relation between a category and a node.
Parameters: **$nodeId** string type is the uuid or path of the document, folder, mail or record **$catId** string type is the text		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
```

```
use openkm\OpenKM;

class ExampleProperty {

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

    private $ws;

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

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

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveCategory();

?>
```

addKeyword

Description:

Method	Return values	Description
addKeyword($nodeId, $keyword)	**void**	Adds a keyword and a node.

Parameters:

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

$keyword string type is the keyword

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two_words" (the character "_" is used for the junction).

Also we suggest you to add keyword in lower case format, because OpenKM is case sensitive.

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleProperty {

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

    private $ws;

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

    public function testAddKeyword() {
        try {
            $this->ws->addKeyword("/okm:root/SDK4PHP/logo.png", "te
            echo 'add keyword';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testAddKeyword();

?>
```

removeKeyword

Description:

Method	Return values	Description
removeKeyword($nodeId, $keyword)	**void**	Removes a keyword from a node.

Parameters:

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

$keyword string type is the keyword

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::
    }

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

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testRemoveKeyword();

?>
```

setEncryption

Description:

Method	Return values	Description
setEncryption($nodeId,	**void**	Marks a document as an encrypted binary data

$cipherName)		into the repository

Parameters:

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

$chipherName 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::
    }

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

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testSetEncryption();
?>
```

unsetEncryption

Description:

Method	Return values	Description
unsetEncryption($nodeId)	**void**	Marks a document as a normal binary data into repository.

Parameters:

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

This method does not perform any kind of uncryption, it simply marks into the database that a document has been uncrypted.

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::
    }

    public function testUnsetEncryption() {
        try {
            $this->ws->unsetEncryption("/okm:root/SDK4PHP/logo.png"
            echo 'unset Encryption';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleProperty = new ExampleProperty();
$exampleProperty->testUnsetEncryption();
```

```
?>
```

setSigned

Description:

Method	Return values	Description
setSigned($nodeId, $signed)	**void**	Marks a document as signed or unsigned binary data into the repository

Parameters:

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

$signed bool type

This method does not perform any kind of digital signature process, it simply marks into the database that a document is signed.

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::
    }

    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
addGroup($nodeId, $grpName)	**void**	Add an empty metadata group to a node.
Parameters: **$nodeId** string type is the uuid or path of the document, folder, mail or record. **$grpName** string type is the grpName should be a valid Metadata group name.		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

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

    private $ws;

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

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

}

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

removeGroup

Description:

Method	Return values	Description
removeGroup($nodeId, $grpName)	**void**	Removes a metadata group of a node.
Parameters: **$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::
    }

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

}

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

getGroups

Description:

Method	Return values	Description
getGroups($nodeId)	**array**	Retrieves a list of metadata groups assigned to a node.
Parameters: **$nodeId** string type is the uuid or path of the document, folder, mail or record.		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

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

    private $ws;

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

    public function testGetGroups() {
        try {
            $propertyGroups = $this->ws->getGroups('/okm:root/SDK4P
            foreach ($propertyGroups as $propertyGroup) {
                var_dump($propertyGroup);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

getAllGroups

Description:

Method	Return values	Description
getAllGroups()	**array**	Retrieves a list of all metadata groups set into the application.

Example:

```
<?php
```

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

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

    private $ws;

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

    public function testGetAllGroups() {
        try {
            $propertyGroups = $this->ws->getAllGroups();
            foreach ($propertyGroups as $propertyGroup) {
                var_dump($propertyGroup);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

getPropertyGroupProperties

Description:

Method	Return values	Description
getPropertyGroupProperties(String nodeId, String grpName)	**List<FormElement>**	Retrieves a list of all metadata group elements and its values of a node.

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.

The method is usually used to display form elements with its values to be shown or changed by used.

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::
    }

    public function testGetPropertyGroupProperties() {
        try {
            $formElements = $this->ws->getPropertyGroupProperties('
            foreach ($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
getPropertyGroupForm($grpName)	**array**	Retrieves a list of all metadata group elements definition.

Parameters:

$grpName string type is the grpName should be a valid Metadata group name.

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExamplePropertyGroup {

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

    private $ws;

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

    public function testGetPropertyGroupForm() {
        try {
            $formElements = $this->ws->getPropertyGroupForm('okg:cc
            foreach ($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

setPropertyGroupProperties($nodeId, $grpName, $formElements)	**void**	Cl

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 ofeList all FormElement, is enought with the formE

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property
                                 "http://www.openkm.com/d
<property-groups>
  <property-group label="Consulting" name="okg:consulting
    <input label="Name" type="text" name="okp:consulting.
    <input label="Date" type="date" name="okp:consulting.
    <checkbox label="Important" name="okp:consulting.impo
    <textarea label="Comment" name="okp:consulting.commer
  </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::
    }
```

```
    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/SDK4PH
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

setPropertyGroupPropertiesSimple

Description:

Method	Return values	
setPropertyGroupPropertiesSimple($nodeId, $grpName, $properties)	**void**	C

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

Is not mandatory set into properties parameter all fields values, is enough with the fields

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property
                                 "http://www.openkm.com/d
<property-groups>
  <property-group label="Consulting" name="okg:consulting
    <input label="Name" type="text" name="okp:consulting.
    <input label="Date" type="date" name="okp:consulting.
```

```
        <checkbox label="Important" name="okp:consulting.impc
        <textarea label="Comment" name="okp:consulting.commer
    </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::
    }

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

            $name = new \openkm\bean\SimplePropertyGroup();
            $name->setName("okp:consulting.name");
            $name->setValue("new value");
            $properties[] = $name;

            $important = new \openkm\bean\SimplePropertyGroup();
            $important->setName("okp:consulting.important");
            $important->setValue("true");
            $properties[] = $important;

            $this->ws->setPropertyGroupPropertiesSimple('/okm:root/
            echo 'updated';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

hasGroup

Description:

Method	Return values	Description
hasGroup($nodeId, $grpName)	**bool**	Returns a boolean that indicate if the node has or not a metadata group.

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::
    }

    public function testSetPropertyGroupPropertiesSimple() {
        try {
            echo 'Have metadata group: ' . $this->ws->hasGroup('/ok
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

Repository samples

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder()	**Folder**	Returns the object Folder of node "/okm:root"

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testGetRoolFolder() {
        try {
            $folders = $this->ws->getRootFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetRoolFolder();
?>
```

getTrashFolder

Description:

Method	Return values	Description
getTrashFolder()	**Folder**	Returns the object Folder of node "/okm:trash/{userId}"

The returned folder will be the user trash folder.

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    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
getTemplatesFolder()	**Folder**	Returns the object Folder of node "/okm:templates"

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    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:

Return

Method	values	Description
getPersonalFolder()	**Folder**	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    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
getMailFolder()	**Folder**	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

For example if the method is executed by "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::
    }

    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
getThesaurusFolder()	**Folder**	Returns the object Folder of node "/okm:thesaurus"

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testGetThesaurusFolder() {
        try {
            $folders = $this->ws->getThesaurusFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetThesaurusFolder();
?>
```

getCategoriesFolder

Description:

Method	Return values	Description

getCategoriesFolder()	**Folder**	Returns the object Folder of node "/okm:categories"

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testGetCategoriesFolder() {
        try {
            $folders = $this->ws->getCategoriesFolder();
            var_dump($folders);
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetCategoriesFolder();
?>
```

purgeTrash

Description:

Method	Return values	Description
purgeTrash()	**void**	Definitively remove from repository all nodes into "/okm:trash/{userId}"

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

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

    public function testPurgeTrash() {
        try {
            $this->ws->purgeTrash();
            echo 'correct';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testPurgeTrash();
?>
```

getUpdateMessage

Description:

Method	Return values	Description

getUpdateMessage()	**string**	Retrieves a message when a new OpenKM release is available.

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

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleRepository {

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

    private $ws;

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

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

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetUpdateMessage();
?>
```

getRepositoryUuid

Description:

Method	Return values	Description

getRepositoryUuid()	**string**	Retrieves 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::
    }

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

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetRepositoryUuid();
?>
```

hasNode

Description:

Method	Return values	Description
hasNode($nodeId)	**bool**	Returns a node that indicates if a node exists or not.
Parameters: **$nodeId** string type is the value of the parameter nodeId can be a valid UUID or path.		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testHasNode() {
        try {
            echo 'Exists node: ' . $this->ws->hasNode('adabdb0f-7ff
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testHasNode();
?>
```

getNodePath

Description:

Method	Return values	Description
getNodePath($uuid)	**string**	Converts a node UUID to path.
Parameters: **$uuid** string type is the uuid of the node		

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testGetNodePath() {
        try {
            var_dump($this->ws->getNodePath('adabdb0f-7ff8-4832-9e4
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetNodePath();
?>
```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid($nodePath)	**string**	Converts a node path to UUID.
Parameters: **$nodePath** string type is the path of the node		

Example:

```
<?php
```

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

    private $ws;

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

    public function testGetNodeUuid() {
        try {
            var_dump($this->ws->getNodeUuid('/okm:root/SDK4PHP/logc
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

$openkm = new OpenKM(); //autoload
$exampleRepository = new ExampleRepository();
$exampleRepository->testGetNodeUuid();
?>
```

getAppVersion

Description:

Method	Return values	Description
getAppVersion()	**AppVersion**	Returns information about 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::
    }

    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	Desc
executeScript($content)	**ScriptExecutionResult**	Executes an sc

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:
    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::
    }

    public function testExecuteScript() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.bsh';
            $scriptExecutionResult = new \openkm\bean\ScriptExecuti
            $scriptExecutionResult = $this->ws->executeScript(file_
            var_dump($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
executeSqlQuery($content)	**SqlQueryResults**	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::
    }

    public function testExecuteSqlQuery() {
        try {
            $fileName = dirname(__FILE__) . '/files/test.sql';
            $sqlQueryResults = new SqlQueryResults();
            $sqlQueryResults = $this->ws->executeSqlQuery(file_get_
            foreach ($sqlQueryResults->getResults() as $sqlQueryRes
                $columns = $sqlQueryResultColumns->getColumns();
                var_dump('uuid: ' . $columns[0] . ' name: ' .  $col
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

```
$exampleRepository = new ExampleRepository();
$exampleRepository->testExecuteSqlQuery();
?>
```

Also the InputStream can be set as:

```
$sql = "SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;";
$sqlQueryResults = $this->ws->executeSqlQuery($sql);
```

executeHqlQuery

Description:

Method	Return values	Description
executeHqlQuery($content)	**HqlQueryResults**	Executes HQL sentences.

Parameters:

$content string type Recommends using file_get_contents — Reads entire file into a string

The testhql.sql script used in the sample below:

```
SELECT uuid, name from NodeBase where name = 'okm:root';
```

The HQL script can only contains a single HQL sentence.

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

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;
use openkm\bean\AppVersion;

class ExampleRepository {

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

```
    private $ws;

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

    public function testExecuteHqlQuery() {
        try {
            $fileName = dirname(__FILE__) . '/files/testhql.sql';
            $hqlQueryResults = new \openkm\bean\HqlQueryResults();
            $hqlQueryResults = $this->ws->executeHqlQuery(file_get_
            foreach ($hqlQueryResults->getResults() as $hqlQueryRes
                var_dump($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

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

Variables	Type	Allow wildcards	Restrictions
domain	**int**	No.	Available values: • QueryParams::DOCUMENT • QueryParams::FOLDER • QueryParams::MAIL • QueryParams::RECORD By default the value is set to QueryParams.DOCUM For searching documents and folders use value: `(QueryParams::DOCUMENT | QueryPa`
author	**string**	No.	Value must be a valid userId.
name	**string**	Yes.	
title	**string**	Yes.	
keywords	**array**	Yes.	

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

mailSubject	**string**	Yes.	Only apply to mail nodes.
mailFrom	**string**	Yes.	Only apply to mail nodes.
mailTo		Yes.	Only apply to mail nodes.
notes		Yes.	

properties	**array**	Yes on almost.	On metadata field values like "date" can not be app
			The map of the properties is composed of pairs:
			('metadata_field_name','metada_field_value")
			For example:
			`$properties = array();`
			`$properties['okp:consulting.name'`

The search operation is done only by AND logic.

Wildcard examples:

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

Methods

findByContent

Description:

Method	Return values	Description
findByContent($content)	**array**	Returns a list of QueryResults filtered by the value of the content parameter.

Parameters:

$content string type

The method only searches among all documents, it 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::
    }

    public function testFindByContent() {
        try {
            $queryResults = $this->ws->findByContent('test*');
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

findByName

Description:

Method	Return values	Description
findByName($name)	**array**	Returns a list of QueryResults filtered by the value of the name parameter.

Parameters:

$name string type

The method only searches among all documents, it not takes in consideration any other kind of nodes.

Example:

```
<?php

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

use openkm\OKMWebServicesFactory;
use openkm\OpenKM;

class ExampleSearch {

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

    private $ws;

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

    public function testFindByName() {
        try {
            $queryResults = $this->ws->findByName('test');
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

findByKeywords

Description:

Method	Return values	Description
findByKeywords($keywords)	**array**	Returns a list of QueryResults filtered by the values of the keywords parameter.

Parameters:

$keywords array type

The method only searches among all documents, it 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::
    }

    public function testFindByKeywords(){
        try {
            $keywords = array();
            $keywords[] = 'php';
            $queryResults = $this->ws->findByKeywords($keywords);
            foreach ($queryResults as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

find

Description:

Method	Return values	Description
find(QueryParams $queryParams)	**array**	Returns a list of QueryResults filtered by the values of the queryParams parameter.

Parameters:

$queryParams QueryParams type

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::
    }

    public function testFind() {
        try {
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryPa
            $queryParams->setFolder("398735af-6282-450e-863c-d00390
            $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

findPaginated(QueryParams $queryParams, $offset, $limit)	**ResultSet**	Returns a list of paginated results filtered by the values of the queryParams parameter.

Parameters:

$queryParams QueryParams type

$offset int type

$limit int type

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 supose 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::
    }

    public function testFindPaginated(){
        try {
            $queryParams = new QueryParams();
            $queryParams->setDomain(QueryParams::DOCUMENT + QueryPa
            $queryParams->setFolder("398735af-6282-450e-863c-d00390
            $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
findSimpleQueryPaginated($statement, $offset, $limit)	**ResultSet**	Returns a list of paginated results filtered by the values of the statement parameter.
Parameters: **$statement** string type		

$offset int type

$limit int type

The **syntax** to use in the statement parameter is the pair '**field:value**'. For example:

- "name:grial" is filtering field name by word grial.

More information about Lucene sintaxis at Lucene query syntax.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
<?php

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

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

class ExampleSearch {
```

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

    private $ws;

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

    public function testFindSimpleQueryPaginated(){
        try {
            $resultSet = $this->ws->findSimpleQueryPaginated('name:
            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
findMoreLikeThis(String uuid, int max)	**ResultSet**	Returns a list of documents that are considered similar by search engine.

Parameters:

$statement string type is the uuid of the Document

$offset int type is the max value is used to limit the number of results returned.

The method can only be used with documents.

Example:

```
<?php

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

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

class ExampleSearch {

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

    private $ws;

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

    public function testFindMoreLikeThis(){
        try {
            $resultSet = $this->ws->findMoreLikeThis("96c44de6-1d0d
            echo "Total results:" . $resultSet->getTotal();
            foreach ($resultSet->getResults() as $queryResult) {
                var_dump($queryResult);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

getKeywordMap

Description:

Method	Return values	Description
getKeywordMap($filter)	**array**	Returns a array of the KeywordMap with its count value filtered by other keywords.

Parameters:

$filter array type is the uuid of the Document

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::
    }

    public function testKeywordMap(){
        try {
            // All keywords without filtering
            echo 'Without filtering';
            $keywordMaps = $this->ws->getKeywordMap();
            foreach ($keywordMaps as $keywordMap) {
                var_dump($keywordMap);
            }
            // Keywords filtered
            echo 'Filtering';
            $filter = array('test','php');
            $keywordMaps = $this->ws->getKeywordMap($filter);
            foreach ($keywordMaps as $keywordMap) {
                var_dump($keywordMap);
            }
        } catch (Exception $e) {
            var_dump($e);
```

```
        }
    }

}

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

getCategorizedDocuments

Description:

Method	Return values	Description
getCategorizedDocuments(String categoryId)	**List<Document>**	Retrieves a list of all documents related with a category.
Parameters: **$categoryId** string type is the uuid or path of the Category		

Example:

```
<?php

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

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

class ExampleSearch {

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

    private $ws;

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

    public function testGetCategorizedDocuments(){
        try {
            $documents = $this->ws->getCategorizedDocuments('abd631
            foreach ($documents as $document) {
                var_dump($document);
            }
```

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

}

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

saveSearch

Method	Return values	Description
saveSearch(QueryParams $params)	**int**	Saves a search parameters and returns the id of the saved search
Parameters: **$params** QueryParams type is the variable queryName of the parameter params, should have to be initialized.		

Example:

```
<?php

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

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

class ExampleSearch {

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

    private $ws;

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

    public function testSaveSearch() {
        try {
            $params = new QueryParams();
            $params->setDomain(QueryParams::DOCUMENT + QueryParams:
```

```
            $params->setName('test*');
            $params->setFolder("398735af-6282-450e-863c-d00390c5bdd
            $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
updateSearch(QueryParams $params)	**void**	Updates a previously saved search parameters.

Parameters:

$params QueryParams type

Only can be updated a saved search created by the same user 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::
    }

    public function testUpdateSearch() {
        try {
            $qpId = 1; // Some valid search id
            $params = $this->ws->getSearch($qpId);
            $params->setName('test*.pdf');
            $this->ws->updateSearch($params);
            echo 'update search';
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

getSearch

Description:

Method	Return values	Description
getSearch($qpId)	**QueryParams**	Gets a saved search parameters.

Parameters:

$qpId int type is the id of the saved search

Only can be retrieved a saved search created by the same user who's executing the method.

Example:

```
<?php

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

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

class ExampleSearch {

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

    private $ws;

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

    public function testGetSearch() {
        try {
            $qpId = 2; // Some valid search id
            $params = $this->ws->getSearch($qpId);
            var_dump($params);
        } catch (Exception $e) {
            var_dump($e);
        }
    }
}

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

getAllSearchs

Description:

Method	Return values	Description
getAllSearchs()	**List<QueryParams>**	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::
    }

    public function testGetAllSearchs() {
        try {
            foreach ($this->ws->getAllSearchs() as $params) {
                var_dump($params);
            }
        } catch (Exception $e) {
            var_dump($e);
        }
    }

}

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

deleteSearch

Description:

Method	Return values	Description
deleteSearch(int qpId)	**void**	Deletes a saved search parameters.

Parameters:

$qpId int type is the id of the saved search

Only can be deleted a saved search created by the same user user who's executing the method.

Example:

```
<?php
```

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

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

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

class ExampleSearch {

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

    private $ws;

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

    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();
?>
```