Documentation for SDK for Java 3.34

Table of Contents

SDK for Java 3.34

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

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

License

SDK for Java is licensed under the terms of the EULA - OpenKM SDK End User License Agreement and published by OpenKM Knowledge Management System S.L.

This program is distributed WITHOUT ANY WARRANTY not even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the EULA - OpenKM SDK End User License Agreement for more details.

Compatibility

SDK for Java version 3.34 should be used:

- **From OpenKM Professional version 7.1.44**

Sample client

You can make use of the OpenKM Maven Repository and the right SDK version. This is a sample pom.xml configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.openkm.sample</groupId>
  <artifactId>sample</artifactId>
  <version>1.0-SNAPSHOT</version>

  <repositories>
    <repository>
      <id>openkm.com</id>
      <name>OpenKM Maven Repository</name>
      <url>https://maven.openkm.com</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>com.openkm</groupId>
      <artifactId>sdk4j</artifactId>
      <version>3.34</version>
    </dependency>
  </dependencies>
</project>
```

Your first class:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Folder fld : ws.folder.getFolderChildren("4f873d10-654e-4d99-a94f-15
                System.out.println("Folder -> " + fld.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Jar sample

If you use "**maven-assembly-plugin**" rather "**maven-shade-plugin**" you will get an error "**missing MessageBodyWriter**" while uploading a new file across the SDK.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.openkm.sample</groupId>
  <artifactId>sample</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <java.compiler>1.8</java.compiler>
    <sdk4j.version>3.34</sdk4j.version>
    <maven-compiler-plugin.version>3.1</maven-compiler-plugin.version>
    <maven-shade-plugin.version>2.4.3</maven-shade-plugin.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <repositories>
    <repository>
      <id>openkm.com</id>
      <name>OpenKM Maven Repository</name>
      <url>https://maven.openkm.com</url>
    </repository>
  </repositories>

  <dependencies>
    <dependency>
      <groupId>com.openkm</groupId>
      <artifactId>sdk4j</artifactId>
      <version>${sdk4j.version}</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>${maven-compiler-plugin.version}</version>
        <configuration>
          <source>${java.compiler}</source>
          <target>${java.compiler}</target>
          <encoding>${project.build.sourceEncoding}</encoding>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-shade-plugin</artifactId>
        <version>${maven-shade-plugin.version}</version>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>shade</goal>
            </goals>
            <configuration>
              <transformers>
                <transformer implementation="org.apache.maven.plugins.shade.resource.
```

```
                  <mainClass>com.openkm.Main</mainClass>
                </transformer>
                <transformer implementation="org.apache.maven.plugins.shade.resource.
                  <resource>META-INF/services/javax.ws.rs.ext.MessageBodyWriter</reso
                </transformer>
              </transformers>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

Basic concepts

Authentication

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

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

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.exception.DatabaseException;
import com.openkm.sdk4j.exception.RepositoryException;
import com.openkm.sdk4j.exception.UnknowException;
import com.openkm.sdk4j.exception.WebserviceException;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getAppVersion());
        } catch (RepositoryException e) {
            e.printStackTrace();
        } catch (DatabaseException e) {
            e.printStackTrace();
        } catch (UnknowException e) {
            e.printStackTrace();
        } catch (WebserviceException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

Accessing API

OpenKM API classes are under com.openkm package, as can shown at this Javadoc API summary.

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

At the moment of writing this page the actual OpenKM version was 7.1.5 what can change with time.

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

OpenKM API classes:

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

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

OKMRecord	Manages records. For example create, move or delete a record.	Yes	RecordImpl.java	BaseRecord.java
OKMRelation	Manages relations between nodes. For example create a relation (parent-child) between two documents.	Yes	RelationImpl.java	BaseRelation.java
OKMRepository	A lot of options related with the repository. For example get the properties of main root node (/okm:root).	Yes	RepositoryImpl.java	BaseRepository.java
OKMReport	Manages reports. For example execute reports.	Yes	ReportImpl.java	BaseReport.java
OKMSearch	Manages search feature. For example manage saved queries or perform a new query to the repository.	Yes	SearchImpl.java	BaseSearch.java
OKMStamp	Manages stamp.	Yes	StampImpl.java	BaseStamp.java
OKMStats	General stats of the repository.	No		
OKMTask	Manages tasks. For	Yes	TaskImpl.java	BaseTask.java

	example create a new task.			
OKMUserConfig	Manages user home configuration.	No		
OKMWorkflow	Manages workflows. For example execute a new workflow.	Yes	WorkflowImpl.java	BaseWorkflow.java

Class Hierarchy

Packages detail:

Name	Description
com.openkm	The **com.openkm.OKMWebservicesFactory** that returns an **com.openkm.OKMWebservices** object which implements all interfaces. `OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);`
com.openkm.sdk4j.bean	Contains all classes result of unmarshalling REST objects.
com.openkm.sdk4j.definition	All interface classes: • com.openkm.sdk4j.definition.BaseActivity • com.openkm.sdk4j.definition.BaseAuth • com.openkm.sdk4j.definition.BaseBookmark • com.openkm.sdk4j.definition.BaseConversion • com.openkm.sdk4j.definition.BaseDashboard • com.openkm.sdk4j.definition.BaseFilePlan • com.openkm.sdk4j.definition.BaseDocument • com.openkm.sdk4j.definition.BaseFolder • com.openkm.sdk4j.definition.BaseMail • com.openkm.sdk4j.definition.BaseNode • com.openkm.sdk4j.definition.BaseNote

	• com.openkm.sdk4j.definition.BaseNotification • com.openkm.sdk4j.definition.BasePlugin • com.openkm.sdk4j.definition.BaseProperty • com.openkm.sdk4j.definition.BasePropertyGroup • com.openkm.sdk4j.definition.BaseRecord • com.openkm.sdk4j.definition.BaseRelation • com.openkm.sdk4j.definition.BaseRepository • com.openkm.sdk4j.definition.BaseSearch • com.openkm.sdk4j.definition.BaseStamp • com.openkm.sdk4j.definition.BaseTask • com.openkm.sdk4j.definition.BaseWorkflow
com.openkm.sdk4j.impl	All interface implementation classes: • com.openkm.sdk4j.impl.ActivityImpl • com.openkm.sdk4j.impl.AuthImpl • com.openkm.sdk4j.impl.BookmarkImpl • com.openkm.sdk4j.impl.ConversionImpl • com.openkm.sdk4j.impl.DashboardImpl • com.openkm.sdk4j.impl.DocumentImpl • com.openkm.sdk4j.impl.FileplanImpl • com.openkm.sdk4j.impl.FolderImpl • com.openkm.sdk4j.impl.MailImpl • com.openkm.sdk4j.impl.NodeImpl • com.openkm.sdk4j.impl.NoteImpl • com.openkm.sdk4j.impl.NotificationImpl • com.openkm.sdk4j.impl.PropertyGroupImpl • com.openkm.sdk4j.impl.PropertyImpl • com.openkm.sdk4j.impl.RecordImpl • com.openkm.sdk4j.impl.RelationImpl • com.openkm.sdk4j.impl.RepositoryImpl • com.openkm.sdk4j.impl.SearchImpl • com.openkm.sdk4j.impl.StampImpl • com.openkm.sdk4j.impl.taskImpl • com.openkm.sdk4j.impl.WorkflowImpl

com.openkm.sdk4j.util	A couple of utilities. The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields.
com.openkm.sdk4j.exception	All exception classes.

Activity samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Activity methods from "**activity**" class as is shown below:

```
ActivityList results = ws.activity.findActivityLog(0, 20, beginDate, endDate, user, "
```

Methods

findActivityLog

Description:

Method	Return values	Description
findActivityLog(int page, int length, Calendar beginDate, Calendar endDate, String user, String action, String item)	**ActivityList**	Returns a list of all activity log by date, user, action and item.
The value of the **item** is the UUID of the node (document, folder, mail or record).		

Example:

```
package com.openkm;

import java.util.Calendar;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Activity;
import com.openkm.sdk4j.bean.ActivityList;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);

            Calendar beginDate = Calendar.getInstance();
            beginDate.add(Calendar.MONTH, -1);
            Calendar endDate = Calendar.getInstance();
            String item = "f84a2e1f-a858-4e53-9c09-36519d903782";

            ActivityList results = ws.activity.findActivityLog(0, 20, beginDate, endD
            for(Activity activity: results.getActivities()) {
             System.out.println(activity);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActivityActions

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (String action : ws.activity.getActivityActions()) {
                System.out.println(action);
            }
```

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

Auth samples

Basics

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

To set READ and WRITE access you should do:

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

To check if you have permission access you should do:

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

Example of uuid:

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

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

```
Map<String, Integer> grants = ws.auth.getGrantedRoles("373bcdd0-c082-4e7b-addd-e10ef8
```

Methods

login

Description:

Method	Return values	Description
login(String user, String password)	String	Login process return authentication token.

Login token by **default** have an **expiration time of 24h**.

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

Each time login method is executed the login token is replaced by newer.

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            System.out.println("Token: " + ws.login(user, password));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

login

Description:

Method	Return values	Description

login(String user, String password, int expiration, boolean restrictIP)	String	Login process return authentication token.

Login token by default have an **expiration** time of variable **expiration value in days**.

When **restrictIP** is true, the token **only will work from the source IP address**.

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

Each time login method is executed the login token is replaced by newer.

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

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

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

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
        int days = 7;
        boolean restrictIp = true;

        try {
            System.out.println("Token: " + ws.login(user, password, days, restrictIp));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

logout

Description:

Method	Return values	Description
logout()	void	Execute logout method in OpenKM side.

Thee authentication token will be reset.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.logout();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGrantedUsersAndRoles

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

```
        try {
            ws.login(username, password);

            GrantedUsersAndRolesItem grants = ws.auth.getGrantedUsersAndRoles("3c68b3
            for (String user : grants.getGrantedUsers().keySet()) {
                System.out.println(user + "->" + grants.getGrantedUsers().get(user));
            }
            for (String role : grants.getGrantedRoles().keySet()) {
                System.out.println(role + "->" + grants.getGrantedRoles().get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGrantedRoles

Description:

Method	Return values	Description
getGrantedRoles(String uuid)	**Map<String, Integer>**	Return the granted roles of a node.

Example:

```
package com.openkm;

import java.util.Map;

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

public class Test {

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

        try {
            ws.login(user, password);
            Map<String, Integer> grants = ws.auth.getGrantedRoles("373bcdd0-c082-4e7b
            for (String role : grants.keySet()) {
                System.out.println(role + "->" + grants.get(role));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGrantedUsers

Description:

Method	Return values	Description

getGrantedUsers(String uuid)	**Map<String, Integer>**	Return the granted users of a node.

Example:

```
package com.openkm;

import java.util.Map;

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

public class Test {

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

        try {
            ws.login(username, password);
            Map<String, Integer> grants = ws.auth.getGrantedUsers("373bcdd0-c082-4e7b
            for (String user : grants.keySet()) {
                System.out.println(user + "->" + grants.get(user));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRoles

Description:

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

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
```

```
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
        try {
            ws.login(username, password);
            for (String role : ws.auth.getRolesByUser("okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRolesByUser

Description:

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

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/OpenKM";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
        try {
            ws.login(username, password);
            for (String role : ws.auth.getRolesByUser("okmAdmin")) {
                System.out.println(role);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUsers

Description:

Method	Return values	Description
getUsers(boolean showAll)	**List<CommonUser>**	Return the list of all the users.

When showAll variable is set to true return all the users - enabled and disabled - otherwise only returns the enabled.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (CommonUser commonUser : ws.auth.getUsers(true)) {
                System.out.println(commonUser);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUser

Description:

Method	Return values	Description
getUser(String userId)	**CommonUser**	Return all user data

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            CommonUser commonUser = ws.auth.getUser("okmAdmin");
            System.out.print(commonUser);
```

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

getUsersByRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (CommonUser commonUser : ws.auth.getUsersByRole("ROLE_ADMIN")) {
                System.out.println(commonUser);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

revokeRole

Description:

Method	Return values	Description
revokeRole(String uuid, String roleId, int permissions, boolean recursive)	**void**	Remove role grant on a node.
The parameter recursive only has sense when the uuid is a folder or a record node.		

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Remove ROLE_USER write grants at the node but not descendants
            ws.auth.revokeRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", P

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

revokeUser

Description:

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

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            // Remove sochoa write grants at the node but not descendants
            ws.auth.revokeUser("373bcdd0-c082-4e7b-addd-e10ef813946e", "sochoa", Perm

            // Remove all okmAdmin grants at the node and descendants
            ws.auth.revokeUser("373bcdd0-c082-4e7b-addd-e10ef813946e", "okmAdmin", Pe
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

grantRole

Description:

Method	Return values	Description
grantRole(String uuid, String role, int permissions, boolean recursive)	**void**	Add role grant on a node.

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Add ROLE_USER write grants at the node but not descendants
            ws.auth.grantRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", Pe

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

```
            e.printStackTrace();
        }
    }
}
```

grantUser

Description:

Method	Return values	Description
grantUser(String uuid, String user, int permissions, boolean recursive)	**void**	Add user grant on a node.
The parameter recursive only has sense when the uuid is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Add sochoa write grants at the node but not descendants
            ws.auth.grantUser("4f873d10-654e-4d99-a94f-15466e30a0f6", "sochoa", Permi

            // Add all okmAdmin grants at the node and descendants
            ws.auth.grantUser("4f873d10-654e-4d99-a94f-15466e30a0f6", "okmAdmin", Per
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

changeSecurity

Description:

Method	Return values	Description

changeSecurity(String uuid, ChangeSecurity changeSecurity)	**void**	Change the security of a node.

Example:

```
package com.openkm;

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

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ChangeSecurity;
import com.openkm.sdk4j.bean.GrantedRole;
import com.openkm.sdk4j.bean.GrantedUser;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);

            List<GrantedUser> guList = new ArrayList<>();
            GrantedUser gu = new GrantedUser();
            gu.setUser("sochoa");
            gu.setPermissions(Permission.ALL_GRANTS);
            guList.add(gu);

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

            ChangeSecurity cs = new ChangeSecurity();
            cs.setGrantedUsersList(guList);
            cs.setGrantedRolesList(grList);
            cs.setRecursive(true);
            ws.auth.changeSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", cs);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

overwriteSecurity

Description:

Method	Return values	Description
overwriteSecurity(String uuid, ChangeSecurity changeSecurity)	**void**	Overwrite the security of a node.

The ChangeSecurity object is used in changeSecurity and overwriteSecurity methods.

Although set values in the variables revokeUsers and revokeRoles of the ChangeSecurity object, these values will not be taken in consideration while overwritten the security.

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            List<GrantedUser> guList = new ArrayList<>();
            GrantedUser gu = new GrantedUser();
            gu.setUser("sochoa");
            gu.setPermissions(Permission.ALL_GRANTS);
            guList.add(gu);

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

            ChangeSecurity cs = new ChangeSecurity();
            cs.setGrantedUsersList(guList);
            cs.setGrantedRolesList(grList);
            cs.setRecursive(true);
            ws.auth.overwriteSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", cs);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasSecurityRecursive

Description:

Method	Return values	Description

hasSecurityRecursive()	Boolean	Check if the user has grants to propagate the security recursively.

The configuration parameter named "**default.security.recursive.role**" is used to indicate the role.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println("hasSecurityRecursive = " + ws.auth.hasSecurityRecursi
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasTaskManagerAdmin

Description:

Method	Return values	Description
hasTaskManagerAdmin()	Boolean	Check if the user is a member of the role task manager admin

The configuration parameter named "**default.task.manager.admin.role**" is used to indicate the role.

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
```

```
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("hasTaskManagerAdmin = " + ws.auth.hasTaskManagerAdmin
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isAdmin

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println("Is Administrator " + ws.auth.isAdmin());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSessionId

Description:

Method	Return values	Description
getSessionId()	**String**	Get http session id.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.auth.getSessionId());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createUser

Description:

Method	Return values	Description
createUser(String userId, String password, String email, String name, boolean active)	**void**	Create a new user.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.auth.createUser("test", "password.2016", "test@mail.com", "User test",
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteUser

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.auth.deleteUser("test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateUser

Description:

Method	Return values	Description
updateUser(String userId, String password, String email, String name, boolean active)	**void**	Update a user.

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

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

createRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

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

deleteRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

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

updateRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

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

assignRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.auth.assignRole("test", "ROLE_USER");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeRole

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

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

```
            e.printStackTrace();
        }
    }
}
```

getProfiles

Description:

Method	Return values	Description
getProfiles(boolean filterByActive)	**List<Profile>**	Return the list of all profiles.

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Profile profile : ws.auth.getProfiles(true)) {
                System.out.println(profile.getName());
                System.out.println(profile.getActive());
                System.out.println(profile.getPrfMisc());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserProfile

Description:

Method	Return values	Description

getUserProfile(String userId)	**Profile**	Return the profile assigned to a user.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Profile profile = ws.auth.getUserProfile("okmAdmin");
            System.out.println(profile.getName());
            System.out.println(profile.getActive());
            System.out.println(profile.getPrfMisc());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserProfile

Description:

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

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

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
```

```
import com.openkm.sdk4j.bean.Profile;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            // Set the profile named "default" to the user
            for (Profile profile : ws.auth.getProfiles(true)) {
                if (profile.getName().equals("default")) {
                    ws.auth.setUserProfile("test", profile.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserToken

Description:

Method	Return values	Description
getUserToken(String userId)	**String**	Return the token to a user.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            String token = ws.auth.getUserToken("okmAdmin");
            System.out.println(token);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserPermissions

Description:

Method	Return values	Description
setUserPermissions(String uuid, String userId, int permissions, boolean recursive)	**void**	Update user permissions on a node.
The parameter recursive only has sense when the uuid is a folder or record node. When parameter recursive is true, the change will be applied to the node and descendants.		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Update permissions of sochoa at the node but not descendants
            ws.auth.setUserPermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "socho

            // Update permissions of okmAdmin at the node and descendants
            ws.auth.setUserPermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "okmAd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRolePermissions

Description:

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Update permissions of ROLE_USER at the node but not descendants
            ws.auth.setRolePermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_
            // Update permissions of ROLE_ADMIN at the node and descendants
            ws.auth.setRolePermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserTenants

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
```

```
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Tenant> tenants = ws.auth.getUserTenants();
            for (Tenant tenant : tenants) {
                System.out.println(tenant);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setUserTenant

Description:

Method	Return values	Description
setUserTenant(long tenantId)	**void**	Change the assigned tenant to a user.

User might have access to several tenants but only have one assigned.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            long tenantId = 1;
            ws.auth.setUserTenant(tenantId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLoginLowercase

Description:

Method	Return values	Description
isLoginLowercase()	**void**	Return true when user is must be set in lowercase in login.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.auth.isLoginLowercase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isPasswordExpired

Description:

Method	Return values	Description
isPasswordExpired()	**void**	True when the password of the user has expired.

> By default the password expriation is disabled. Take a look at the configuration parameter named "**user.password.expiration**" in the Security configuration parameters section to enable.

Example:

```
package com.openkm;

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

public class Test {

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

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

        try {
            ws.login(user, password);
            System.out.println("Password expired: " + ws.auth.isPasswordExpired());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getToken

Description:

Method	Return values	Description
getToken()	void	Get a new authentication token

Authentication token expires after 24 hours or later (it depends on how was created). This method helps in refreshing the current token.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.getToken();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Autocad samples

Basics

Example of uuid:

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

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Autocad methods from "**autocad**" class as is shown below:

```
List<XRef> xrefList = ws.autocad.getXrefs("1be884f4-5758-4985-94d1-f18bfe004db8");
```

Methods

getXRefs

Description:

Method	Return values	Description
getXRefs(String uuid)	**List<XRef>**	Returns a list of the document references of a Autocad document

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(username, password);

            for (XRef xRef : ws.autocad.getXrefs("5ce6dd37-7472-404a-878e-274005a2d6d
                System.out.println(xRef);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

refresh

Description:

Method	Return values	Description
refresh(String uuid)	**void**	refresh references of an Autocad document

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(username, password);

            ws.autocad.refresh("5ce6dd37-7472-404a-878e-274005a2d6db");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Bookmark samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

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

Methods

getUserBookmarks

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

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

        try {
            ws.login(user, password);
            for (Bookmark bookmark : ws.bookmark.getUserBookmarks()) {
                System.out.println(bookmark);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createBookmark

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.bookmark.createBookmark("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameBookmark

Description:

Method	Return values	Description
renameBookmark(int bookmarkId, String name)	**void**	Rename a bookmark

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            int bookmarkId = 1;
            ws.bookmark.renameBookmark(bookmarkId, "set bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteBookmark

Description:

Method	Return values	Description
deleteBookmark(int bookmarkId)	**void**	Delete a bookmark.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            int bookmarkId = 1;
            ws.bookmark.deleteBookmark(bookmarkId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Conversion samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

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

Methods

doc2pdf

Description:

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

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.docx");
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.pdf");
            InputStream convertedStream = ws.conversion.doc2pdf(is, "test.docx");
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

imageConvert

Description:

Method	Return values	Description
imageConvert(InputStream is, String fileName, String params, String dstMimeType)	**InputStream**	Retrieve the uploaded image with transformation.

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

The parameter dstMimeType is the expected document MIME TYPE result.

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

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

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

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

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

When building your integrations, we suggest installing ImageMagic software locally, and check your image transformations first from your command line.

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.jpg");
            InputStream convertedStream = ws.conversion.imageConvert(is, "test.png",
            IOUtils.copy(convertedStream, fos);
            is.close();
            convertedStream.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

html2pdf

Description:

Method	Return values	Description
html2pdf(String url)	**InputStream**	Retrieve the PDF of an URL.

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

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            FileOutputStream fos = new FileOutputStream("/home/openkm/out.pdf");
            InputStream is = ws.conversion.html2pdf("https://www.openkm.com/es/");
            IOUtils.copy(is, fos);
            is.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

doc2txt

Description:

Method	Return values	Description
doc2txt(InputStream is, String fileName)	**String**	Extracts the text from the upload document.

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

Example:

```
package com.openkm;

import java.io.FileInputStream;
```

```
import java.io.InputStream;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.docx");
            System.out.println(ws.conversion.doc2txt(is, "test.docx"));
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

img2txt

Description:

Method	Return values	Description
img2txt(InputStream is, String fileName)	**String**	Extracts the text from the uploaded image.

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

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            System.out.println(ws.conversion.img2txt(is, "test.png"));
```

```
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

barcode2txt

Description:

Method	Return values	Description
barcode2txt(InputStream is, String fileName)	**String**	Extracts the barcode text from the uploaded image.

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

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.png");
            System.out.println(ws.conversion.barcode2txt(is, "test.png"));
            is.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Dashboard samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

```
ws.dashboard.getUserCheckedOutDocuments(0, 5);
```

Methods

getUserCheckedOutDocuments

Description:

Method	Return values	Description
getUserCheckedOutDocuments(int offset, int limit)	**DashboardResultSet**	Returns a list of the documents in edition by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserCheckedOutDocuments(0,
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastModifiedDocuments

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastModifiedDocuments(
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedDocuments

Description:

Method	Return values	Description
getUserLockedDocuments(int offset, int	**DashboardResultSet**	Returns a list of the documents locked by the

limit)		user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedDocuments(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLockedRecords

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedRecords(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

getUserLockekFolders

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedFolders(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
```

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

getUserLockedMails

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
```

```
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedMails(0, 5);
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedDocuments

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedDocuments(0,
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedFolders

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedFolders(0, 5
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSubscribedRecords

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedRecords(0, 5
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedDocuments

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedDocuments(0
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastDocumentsNotesCreated

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastDocumentsNotesCrea
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedFolders

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedFolders(0,
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastFoldersNotesCreated

Description:

Method	Return values	Description
getUserLastFoldersNotesCreated(int offset, int limit)	**DashboardResultSet**	Returns a list of the last folders notes created by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastFoldersNotesCreate
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastCreatedRecords

Description:

Method	Return values	Description

getUserLastCreatedRecords(int offset, int limit)	**DashboardResultSet**	Returns a list of the last records created by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedRecords(0,
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastRecordsNotesCreated

Description:

Method	Return values	Description
getUserLastRecordsNotesCreated(int offset, int limit)	**DashboardResultSet**	Returns a list of the last records notes created by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastRecordsNotesCreate
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
```

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

getUserLastDownloadedDocuments

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

```
        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastDownloadedDocument
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMails

Description:

Method	Return values	Description
getUserLastImportedMails(int offset, int limit)	**DashboardResultSet**	Returns a list of the last mails imported by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMails(0, 5
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastMailsNotesCreated

Description:

Method	Return values	Description
getUserLastMailsNotesCreated(int offset, int limit)	**DashboardResultSet**	Returns a list of the last mails notes created by the user.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastMailsNotesCreated(
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserLastImportedMailAttachments

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.DashboardResult;
import com.openkm.sdk4j.bean.DashboardResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMailAttachm
            System.out.println("Total: " + resultSet.getTotal());
            for (DashboardResult dashboardResult : resultSet.getResults()) {
                System.out.println(dashboardResult.getNode());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUserSearches

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

```
        try {
            ws.login(user, password);
            List<QueryParams> results = ws.dashboard.getUserSearches();
            for (QueryParams userSearch : results) {
                System.out.println(userSearch);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findUserSearches

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            long qpId = 1;
            List<DashboardNodeResult> results = ws.dashboard.findUserSearches(qpId);
            for (DashboardNodeResult nodeResult : results) {
                System.out.println(nodeResult);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Document samples

Basics

Example of uuid:

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

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

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

Methods

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, String name, InputStream is)	**Document**	Creates a new document. Return an object Document with the properties of the created document.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            Document doc = ws.document.createDocument("1be884f4-5758-4985-94d1-f18bfe
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, String name, InputStream is, long nodeClass)	**Document**	Creates a new document with nodeClass. Return an object Document with the properties of the created document.
The values of the uuid parameter should be a folder or record node UUID. The nodeClass parameter should be a valid bussiness type (serie) value. A nodeClass with value 0 means there's no business type (serie) selected.		

Example:

```
package com.openkm;

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

```
import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            long nodeClass = 0;
            Document doc = ws.document.createDocument("1be884f4-5758-4985-94d1-f18bfe
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteDocument

Description:

Method	Return values	Description
deleteDocument(String uuid)	**void**	Delete a document.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.deleteDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getDocumentProperties

Description:

Method	Return values	Description
getDocumentProperties(String uuid)	**Document**	Return the document properties.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.getDocumentProperties("1ec49da9-1746-4875-
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	Return values	Description
getContent(String uuid)	**InputStream**	Retrieve document content - binary data - of the actual document versio

In case you sent the file across a Servlet response we suggest set the content length with:

```
Document doc = ws.getDocumentProperties("1ec49da9-1746-4875-ae32-9281d7303a
response.setContentLength(new Long(doc.getActualVersion().getSize()).intVal
```

We've found wrong size problems while using:

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

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            OutputStream fos = new FileOutputStream("/home/openkm/logo.png");
            InputStream is = ws.document.getContent("1ec49da9-1746-4875-ae32-9281d730
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContentByVersion

Description:

Method	Return values	Description
getContentByVersion(String uuid, String versionName)	**InputStream**	Retrieves a document content (binary data) from a sp document version.

In case you sent the file across a Servlet response we suggest set the content length with:

```
Document doc = ws.getDocumentProperties("1ec49da9-1746-4875-ae32-9281d7303a
response.setContentLength(new Long(doc.getActualVersion().getSize()).intVal
```

We've found wrong size problems while using:

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

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            OutputStream fos = new FileOutputStream("/home/openkm/logo.png");
            InputStream is = ws.document.getContentByVersion("/okm:root/logo.png", "1
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentChildren

Description:

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

Example:

```
package com.openkm;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Document doc : ws.document.getDocumentChildren("1be884f4-5758-4985-9
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameDocument

Description:

Method	Return values	Description
renameDocument(String uuid, String newName)	**Document**	Changes the name of a document.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Document doc = ws.document.renameDocument("1ec49da9-1746-4875-ae32-9281d7
            System.out.print(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setProperties

Description:

Method	Return values	Description
setProperties(String uuid, String title, String description, String lang, List<String> keywords, List<String> categories)	**void**	Changes some document properties.

The parameter lang must be ISO 691-1 compliant.

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

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            List<String> keywords = new ArrayList<>();
            keywords.add("test");
            keywords.add("invoice");

            List<String> categories = new ArrayList<>();
            categories.add("58c9b25f-d83e-4006-bd78-e26d7c6fb648");

            ws.document.setProperties("1ec49da9-1746-4875-ae32-9281d7303a62", "new ti
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setLanguage

Description:

Method	Return values	Description

setLanguage(String uuid, String lang)	**void**	Sets the document language.

The parameter lang must be ISO 691-1 compliant.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.setLanguage("1ec49da9-1746-4875-ae32-9281d7303a62", "en");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDocumentTitle

Description:

Method	Return values	Description
setDocumentTitle(String uuid, String title)	**void**	Sets the document title.

Example:

```
package com.openkm;

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

public class Test {

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

```
        try {
            ws.login(user, password);
            ws.document.setDocumentTitle("1ec49da9-1746-4875-ae32-9281d7303a62", "Som
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkout

Description:

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

Only one user can modify a document at a time.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.checkout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point the document is locked for other users except for the us
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

cancelCheckout

Description:

Method	Return values	Description

cancelCheckout(String uuid)	**void**	Cancels a document edition.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // At this point the document is locked for other users except for the us
            ws.document.cancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceCancelCheckout

Description:

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

This method allows to cancel edition on any document.

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

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

Example:

```
package com.openkm;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            // At this point the document is locked for other users except for the us
            ws.document.forceCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isCheckedOut

Description:

Method	Return values	Description
isCheckedOut(String docId)	**Boolean**	Returns a boolean that indicates if the document is on edition or not.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println("Is the document checkout:" + ws.document.isCheckedOut
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkin

Description:

Method	Return values	Description
checkin(String uuid, InputStream is, String comment)	**Version**	Updates a document with a new version and returns an object with new Version values.

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

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.checkin("1ec49da9-1746-4875-ae32-9281d7303a62", is, "optional
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkin

Description:

Method	Return values	Description
checkin(String docId, InputStream is, String comment, int increment)	**Version**	Updates a document with a new version and returns an object with new Version values.

The value of increment variable must be 1 or greater.

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

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

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.checkin("1ec49da9-1746-4875-ae32-9281d7303a62", is, "optional
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lockDocument

Description:

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

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

A locked document cannot be modified by other users.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.lockDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlockDocument

Description:

Method	Return values	Description
unlockDocument(String uuid)	**void**	Unlocks a locked document.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.unlockDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlockDocument

Description:

Method	Return values	Description
forceUnlockDocument(String uuid)	**void**	Unlocks a locked document.

This method allows to unlock any locked document.

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.forceUnlockDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isLocked

Description:

Method	Return values	Description
isLocked(String uuid)	**Boolean**	Returns a boolean that indicates if the document is locked or not.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println("Is document locked:" + ws.document.isLocked("1ec49da9
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLockInfo

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.getLockInfo("1ec49da9-1746-4875-ae32-9281d
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeDocument

Description:

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.purgeDocument("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveDocument

Description:

Method	Return values	Description
moveDocument(String uuid, String dstId)	**void**	Move document into some folder or record.

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

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.moveDocument("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyDocument

Description:

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

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

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
```

```
            ws.login(user, password);
            ws.document.copyDocument("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistorySize

Description:

Method	Return values	Description
getVersionHistorySize(String uuid)	**long**	Returns the sum in bytes of all documents into documents history.

Example:

```
package com.openkm;

import java.util.Locale;

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

public class Test {

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

        try {
            ws.login(user, password);
            String[] UNITS = new String[]{"B", "KB", "MB", "GB", "TB", "PB", "EB"};
            long bytes = ws.document.getVersionHistorySize("9ed5c7b1-9314-4479-8f80-f
            String value = "";

            for (int i = 6; i > 0; i--) {
                double step = Math.pow(1024, i);
                if (bytes > step) {
                    value = String.format(Locale.ROOT, "%3.1f %s", bytes / step, UNIT
                }
            }

            if (value.isEmpty()) {
                value = Long.toString(bytes) + " " + UNITS[0];
            }

            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValidDocument

Description:

Method	Return values	Description
isValidDocument(String docId)	**Boolean**	Returns a boolean that indicates if the node is a document or not.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.isValidDocument("9ed5c7b1-9314-4479-8f80-f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentPath

Description:

Method	Return values	Description
getDocumentPath(String uuid)	**String**	Converts a document UUID to document path.

Example:

```
package com.openkm;

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

public class Test {

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

```
        try {
            ws.login(user, password);
            System.out.println(ws.document.getDocumentPath("9ed5c7b1-9314-4479-8f80-f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDetectedLanguages

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (String lang : ws.document.getDetectedLanguages()) {
                System.out.println(lang);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedDocumentCopy

Description:

Method	Return values	Description
extendedDocumentCopy(String nodeId, String dstId, String name, boolean categories, boolean keywords, boolean propertyGroups, boolean notes,	**void**	Copies a document with associated data into some

boolean security)		folder or record.

The values of the nodeId parameter should be a Document UUID.

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

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

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

Additional:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Document document = ws.document.extendedDocumentCopy("055b5206-35d0-4351-
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getExtractedText

Description:

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

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

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

Althoughthis restriction, from API - only administrators users - can do database queries to retrieve this information. Check Repository samples for accessing database level.

More information at Statistics.

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);

            InputStream is = new FileInputStream("/home/openkm/test.pdf");
            ws.document.setExtractedText("301de803-f4ae-48f1-8505-e83b26716956", is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setExtractedText

Description:

Method	Return values	Description
setExtractedText(String uuid, InputStream is)	**void**	Set extracted text

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            String text = ws.document.getExtractedText("46762f90-82c6-4886-8d21-ad301
            System.out.println(text);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getThumbnail

Description:

Method	Return values	Description
getThumbnail(String uuid, ThumbnailType type)	**InputStream**	Returns thumbnail image data.

Available types:

- ThumbnailType.THUMBNAIL_PROPERTIES (shown in properties view)
- ThumbnailType.THUMBNAIL_LIGHTBOX (shown in light box)
- ThumbnailType.THUMBNAIL_SEARCH (shown in search view)

Each thumbnail type has its own image dimensions.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            OutputStream fos = new FileOutputStream("/home/openkm/invoice.png");
            InputStream is = ws.document.getThumbnail("46762f90-82c6-4886-8d21-ad3017
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createDocumentFromTemplate

Description:

Method	Return values	Description
createDocumentFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map<String, String> properties)	**Document**	Creates a new document from the template and returns an object Document.

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

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

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

For more information about Templates and metadata check: Creating templates Creating templates

Additional:

- When category parameter is true the original values of the categories will be copied.

- When keywords parameter is true the original values of the keywords will be copied.
- When notes parameter is true the original values of the notes will be copied.

Example:

The example below is based on Creating PDF template sample.

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";

        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            Map<String, String> properties = new HashMap<>();
            // okg:tpl
            properties.put("okp:tpl.name", "sdk name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
            properties.put("okp:tpl.language", "[ \"java\" ]");

            // Property okg:technology
            properties.put("okp:technology.comment", "sdk name");

            Document document = ws.document.createDocumentFromTemplate("5e72dec8-2409
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateDocumentFromTemplate

Description:

Method	Return values	Description

updateDocumentFromTemplate(String uuid, String dstId, Map<String, String> properties)	**Document**	Updates a document previously created from the template and returns an object Document.

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

The dstId is the document to updated UUID.

This method only has sense when the template uses metadata groups to fill in fields.

For more information about Templates and metadata take a look at Creating templates .

Example:

The example below is based on Creating PDF template sample.

```
package com.openkm;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

public class Test {

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

        try {
            ws.login(user, password);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:tpl.name", "update Some name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
            properties.put("okp:tpl.language", "[ \"python\" ]");

            ws.document.updateDocumentFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAnnotations

Description:

Method	Return values	Description
getAnnotations(String uuid, String versionName)	**String**	Return the document annotations of some document version.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.getAnnotations("9ed5c7b1-9314-4479-8f80-fd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDifferences

Description:

Method	Return values	Description
getDifferences(String uuid, String versionName1, String versionName2)	**InputStream**	Return a PDF document with the differences between two document versions.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {
```

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

        try {
            ws.login(user, password);
            InputStream is = ws.document.getDifferences("46762f90-82c6-4886-8d21-ad30
            FileOutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/out.
            IOUtils.copy(is, fos);
            is.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCheckedOut

Description:

Method	Return values	Description
getCheckedOut()	**List<Document>**	Return a list of documents checkout by the user.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Document doc : ws.document.getCheckedOut()) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createDocument

Description:

Method	Return values	Description
createDocument(String uuid, File file)	**Document**	Creates a new document and returns as a result an object Document with the properties of the created document.
The values of the **uuid** parameter should be a folder or record node UUID.		

Example:

```
package com.openkm;

import java.io.File;

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

public class Test {

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

        try {
            ws.login(user, password);
            File file = new File("/home/gnujavasergio/okm/logo.png");
            Document doc = ws.document.createDocument("151f3a54-f370-47d6-801a-d20fae
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDocumentNodeClass

Description:

Method	Return values	Description
setDocumentNodeClass(String uuid, long ncId)	**void**	Set the NodeClass.

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            long ncId = 2;
            ws.document.setDocumentNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b",
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDocumentDispositionStage

Description:

Method	Return values	Description
setDocumentDispositionStage(String uuid, long stage)	**void**	Set the disposition stage

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            long stage = 1;
            ws.document.setDocumentDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDocumentDescription

Description:

Method	Return values	Description

setDocumentDescription(String uuid, String description)	**void**	Set a description.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.setDocumentDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b"
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizardDocument

Description:

Method	Return values	Description
createWizardDocument(String uuid, String name, long nodeClass, InputStream is)	**WizardNode**	Create a new document with wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```
package com.openkm;
```

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            WizardNode wn = ws.document.createWizardDocument("1f323e88-64ee-4f57-91e2
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isOCRDataCaptureSupported

Description:

Method	Return values	Description
isOCRDataCaptureSupported(String uuid)	**boolean**	Check if the node supports an OCR data capture.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            if (ws.document.isOCRDataCaptureSupported("b2f88679-e3fd-4f97-bf0e-abf76f
                System.out.println("Yes supported OCR Data capture");
            } else {
                System.out.println("No supported OCR Data capture");
            }
```

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

recognize

Description:

Method	Return values	Description
recognize(String uuid)	**OCRRecognise**	Return an OCRRecognise object.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            OCRRecognise ocr = ws.document.recognize("b2f88679-e3fd-4f97-bf0e-abf76f9
            System.out.println(ocr);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

captureData

Description:

Method	Return values	Description
captureData(String uuid, long templateId)	**void**	Proceed to the OCR data capture using OCR template identified by templateId.

Example:

```
package com.openkm;

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

public class Test {

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

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

getNumberOfPages

Description:

Method	Return values	Description
getNumberOfPages(String uuid)	**int**	Get the number of pages of a document.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println("The number of pages is : " + ws.document.getNumberOfP
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPageAsImage

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.getPageAsImage("3fe350e2-69e8-4681-a97c-6a
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getLiveEditRestrictedMimeTypes

Description:

Method	Return values	Description
getLiveEditRestrictedMimeTypes()	**List<String>**	Return a list of mymetypes what can not be used with liveedit feature.

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {
```

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

        try {
            ws.login(user, password);
            List<String> mymeTypes = ws.document.getLiveEditRestrictedMimeTypes();
            for (String mimeType : mymeTypes) {
                System.out.println(mimeType);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

liveEditCheckin

Description:

Method	Return values	Description
liveEditCheckin(String uuid, String comment, int increment)	**void**	It does a live edit checkin.

The method should be used only when the document has been edited by live edit.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.liveEditCheckin("8b9a32c8-db65-41c8-a2a0-af03761f60b6", "comm
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

mergePdf

Description:

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

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            List<String> uuids = new ArrayList<>();
            uuids.add("3fe350e2-69e8-4681-a97c-6ac4ba6c1524");
            uuids.add("8b9a32c8-db65-41c8-a2a0-af03761f60b6");
            String fldUuid = "7213e597-c147-46c9-a90b-ac3966b3114b";
            String uuid = ws.document.mergePdf(fldUuid, "MergePdf.pdf", uuids);
            Document pdfDoc = ws.document.getDocumentProperties(uuid);
            System.out.println(pdfDoc.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

liveEditSetContent

Description:

Method	Return values	Description
liveEditSetContent(String uuid, InputStream is	**Void**	Update the content of the document edited with live edit feature.

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.document.liveEditSetContent("1ec49da9-1746-4875-ae32-9281d7303a62", is
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isAttachment

Description:

Method	Return values	Description
isAttachment(String docId)	**Boolean**	Returns a boolean that indicates if the node is a mail attachment or not.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
```

```
            ws.login(user, password);
            System.out.println(ws.document.isAttachment("9ed5c7b1-9314-4479-8f80-fd8e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isConvertibleToPDF

Description:

Method	Return values	Description
isConvertibleToPDF(String docId)	**Boolean**	Returns a boolean that indicates if the node is convertible to PDF or not.

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.document.isConvertibleToPDF("9ed5c7b1-9314-4479-8f8
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceLockDocument

Description:

Method	Return values	Description

forceLockDocument(String uuid)	**LockInfo**	Locks a document and returns an object with the Lock information.

This method allows to lock any document.

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            LockInfo lockInfo = ws.document.forceLockDocument("1ec49da9-1746-4875-ae3
            System.out.println("Author:" + lockInfo.getOwner());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDocumentPdf

Description:

Method	Return values	Description
getDocumentPdf(String uuid)	**InputStream**	Returns a PDF of the document.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            Document doc = ws.document.getDocumentProperties("7f6c5c0b-a66b-4782-9595
            InputStream is = ws.document.getDocumentPdf(doc.getUuid());
            OutputStream fos = new FileOutputStream("/home/openkm/book.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveDocumentAsPdf

Description:

Method	Return values	Description
saveDocumentAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)	**Document**	Save the document as PDF in the OpenKM repository.

The value of the **uuid** parameter should be a Document UUID.

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

Additional:

- When the propertyGroups parameter is true the metadata groups of the source are copied to the target.
- When the security parameter is true the security of the source is copied to the target.

Example:

```
package com.openkm;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            Document doc = ws.document.getDocumentProperties("7f6c5c0b-a66b-4782-9595
            Document docPdf = ws.document.saveDocumentAsPdf(doc.getUuid(), "book.pdf"
            System.out.println("Document pdf: " + docPdf.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

webPageImport

Description:

Method	Return values	Description
webPageImport(String uuid, String url)	**void**	Save a document as PDF with the content of the web pag.
The value of the **uuid** parameter should be a folder or record node UUID. The value of the **url** parameter is any web page		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.webPageImport("271cc3aa-218e-4574-8065-c34c704f4a20", "https:
```

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

setIndexable

Description:

Method	Return values	Description
setIndexable(String uuid, boolean enabled)	**void**	Sets the document indexable.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.document.setIndexable("19938640-fd34-421c-90c8-6b435e4b7d79", true);
            Document document = ws.document.getDocumentProperties("19938640-fd34-421c
            System.out.println("Document indexable:" + document.isIndexable());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

FilePlan samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

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

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

Methods

getRootSections

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClassSectionDefinition> list = ws.filePlan.getRootSections();
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRootSectionsFilteredBySecurity

Description:

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

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            List<NodeClassSectionDefinition> list = ws.filePlan.getRootSectionsFilte
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getChildrenSections

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            int parentId = 1;
            List<NodeClassSectionDefinition> list = ws.filePlan.getChildrenSections(p
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenSectionsFilteredBySecurity

Description:

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

		permissions.

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            int parentId = 4;
            List<NodeClassSectionDefinition> list = ws.filePlan.getChildrenSectionsFi
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenClasses

Description:

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

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            int sectionId = 3;
            List<NodeClass> list = ws.filePlan.getChildrenClasses(sectionId);
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenClassesFilteredBySecurity

Description:

Method	Return values	Description
getChildrenClassesFilteredBySecurity(long sectionId, int permissions)	**List<NodeClass>**	Returns a list of the NodeClass by section and permissions.

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
```

```
            ws.login(user, password);
            int sectionId = 3;
            List<NodeClass> list = ws.filePlan.getChildrenClassesFilteredBySecurity(s
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findNodeClassByPk

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            int id = 4;
            NodeClass nodeClass = ws.filePlan.findNodeClassByPk(id);
            System.out.println(nodeClass);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findElectronicRecordClasses

Description:

Method	Return values	Description
findElectronicRecordClasses()	**List<NodeClass>**	Returns a list of the NodeClass what are Elecrronic Records.

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<NodeClass> list = ws.filePlan.findElectronicRecordClasses();
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findElectronicRecordClassesFilteredBySecurity

Description:

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

> The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClass> list = ws.filePlan.findElectronicRecordClassesFilteredBySe
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findFilteredByCodeOrNameFilteredBySecurity

Description:

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

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClass;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClass> list = ws.filePlan.findFilteredByCodeOrNameFilteredBySecu
```

```
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSectionFiltered

Description:

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

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that are changing or getting security grants.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeClassSectionDefinition;
import com.openkm.sdk4j.bean.Permission;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8090/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NodeClassSectionDefinition> list = ws.filePlan.findSectionFiltered("
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodeClassBreadcrumb

Description:

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

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            int sectionId = 3;
            List<NodeClassSectionDefinition> list = ws.filePlan.getNodeClassBreadcrumb
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findAllNodeClasses

Description:

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

Example:

```
package com.openkm;
```

```
import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<NodeClass> list = ws.filePlan.findAllNodeClasses();
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRootSectionsIdWithChildren

Description:

Method	Return values	Description
getRootSectionsIdWithChildren()	**List<Long>**	Returns a list of the section ids what have childs and parent is root.

Example:

```
package com.openkm;

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

import java.util.List;

public class Test {

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

        try {
            ws.login(user, password);
            List<Long> list = ws.filePlan.getRootSectionsIdWithChildren();
            for (long id : list) {
                System.out.println(id);
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getChildrenSectionsIdByTenantWithChildren

Description:

Method	Return values	Description
getChildrenSectionsIdByTenantWithChildren(long parendId)	**List<Long>**	Returns a list of the section ids what have childs with parent equals to parentId and filtered by tenant.

Example:

```
package com.openkm;

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

import java.util.List;

public class Test {

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

        try {
            ws.login(user, password);
            long parentId = 1;
            List<Long> list = ws.filePlan.getChildrenSectionsIdByTenantWithChildren(p
            for (long id : list) {
                System.out.println(id);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Folder samples

Basics

Example of uuid:

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

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Folder methods from "**folder**" class as is shown below:

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

Methods

createFolder

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Folder folder = ws.folder.createFolder("1be884f4-5758-4985-94d1-f18bfe004
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFolderProperties

Description:

Method	Return values	Description
getFolderProperties(String uuid)	**Folder**	Return the folder properties.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.folder.getFolderProperties("76f4155e-42af-4be4-9a1c
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteFolder

Description:

Method	Return values	Description
deleteFolder(String fldId)	**void**	Delete a folder.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.deleteFolder("76f4155e-42af-4be4-9a1c-756497616fb6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameFolder

Description:

Method	Return values	Description
renameFolder(String fldId, String newName)	**void**	Rename a folder.

Example:

```
package com.openkm;

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

public class Test {

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

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

```
            // Exists folder /okm:root/test
            ws.folder.renameFolder("91264771-02b9-471d-b7a3-ba8cc49a10dd", "renamedFo
            // Folder has renamed to /okm:root/renamedFolder
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveFolder

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Exists folder /okm:root/test
            ws.folder.moveFolder("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae
            // Folder has moved to /okm:root/tmp/test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFolderChildren

Description:

Method	Return values	Description
getFolderChildren(String uuid)	**List<Folder>**	Returns a list of all folders which their parent is fldId.
The parameter uuid can be a folder or a record node.		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Folder fld : ws.folder.getFolderChildren("1be884f4-5758-4985-94d1-f1
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValidFolder

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Return false
            ws.folder.isValidFolder("8cd1e072-8595-4dd3-b121-41d622c43f08");

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

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

getFolderPath

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.folder.getFolderPath("1be884f4-5758-4985-94d1-f18bf
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyFolder

Description:

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

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

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.copyFolder("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedFolderCopy

Description:

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

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

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

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

Additional:

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

- When the notes parameter is true the original values of the notes will be copied.
- When the security parameter is true the original value of the security will be copied.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Folder folder = ws.folder.extendedFolderCopy("ada67d44-b081-4b23-bdc1-741
                    "new name folder", true, true, true, true, true);
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContentInfo

Description:

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

The ContentInfo object retrives information about:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.folder.getContentInfo("ada67d44-b081-4b23-bdc1-7418
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeFolder

Description:

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.purgeFolder("ada67d44-b081-4b23-bdc1-74181cafbc5d");
```

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

setStyle

Description:

Method	Return values	Description
setStyle(String uuid, long styleId)	**void**	Set the folder style.

More information at Folder style.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.setStyle("ada67d44-b081-4b23-bdc1-74181cafbc5d", 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createMissingFolders

Description:

Method	Return values	Description
createMissingFolders(String fldPath)	**String**	Create missing folders.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.createMissingFolders("/okm:root/missingfld1/missingfld2/missing
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setFolderDescription

Description:

Method	Return values	Description
setFolderDescription(String uuid, String description)	**void**	Set a description.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.folder.setFolderDescription("4c195453-246b-4ce9-86ba-b84e68d1f284", "s
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFolderFromTemplate

Description:

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

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

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

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

Additional:

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

Example:

```
package com.openkm.example;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Folder;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FolderExample {

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

        try {
            ws.login(user, password);

            // createFolderFromTemplate
            Map<String, String> properties = new HashMap<>();
            // okg:tpl
```

```
            properties.put("okp:tpl.name", "sdk name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
            properties.put("okp:tpl.language", "[ \"java\" ]");

            // Property okg:technology
            properties.put("okp:technology.comment", "sdk name");
            Folder folder = ws.folder.createFolderFromTemplate("47c70047-2924-483c-bc
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Import samples

Basics

We suggest executing the methods below for huge import or for simple creation cases.

These methods execute fewer steps in the background either what is described in Document samples and Folder samples. That means you will get extra performance in the action because there are executed less logic in the server side.

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Import methods from "**quickImport**" class as is shown below:

```
ws.quickImport.importDocument("1be884f4-5758-4985-94d1-f18bfe004db8", file);
```

Methods

importDocument

Description:

Method	Return values	Description
importDocument(String uuid, File file)	**String**	Creates a new document. Return an the UUID of the Document.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```
package com.openkm;

import java.io.File;

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

public class Test {

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

        try {
            ws.login(user, password);
            File file = new File("/home/gnujavasergio/okm/logo.png");
            String uuid = ws.quickImport.importDocument("1be884f4-5758-4985-94d1-f18b
            System.out.println(uuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importFolder

Description:

Method	Return values	Description
importFolder(String uuid, String name)	**String**	Creates a new folder. Return an the UUID of the folder.
The values of the uuid parameter should be a folder or record node UUID.		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            String uuid = ws.quickImport.importFolder("1be884f4-5758-4985-94d1-f18bfe
```

```
            System.out.println(uuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Mail samples

Basics

Example of uuid:

- Using UUID -> "**064ff51a-b815-4f48-a096-b4946876784f**";

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Mail methods from " **mail**" class as is shown below:

```
ws.mail.getMailProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbced")
```

Methods

getMailProperties

Description:

Method	Return values	Description
getMailProperties(String uuid)	**Mail**	Returns the mail properties.

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.mail.getMailProperties("05d9b8e3-f9c1-4ace-9007-afd
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteMail

Description:

Method	Return values	Description
deleteMail(String uuid)	**void**	Deletes a mail.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.deleteMail("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeMail

Description:

Method	Return values	Description
purgeMail(String uuid)	**void**	Mail is definitely removed from repository.

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.purgeMail("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameMail

Description:

Method	Return values	Description
renameMail(String uuid, String newName)	**void**	Rename a mail.

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

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.renameMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveMail

Description:

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.moveMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyMail

Description:

Method	Return values	Description

public void copyMail(String uuid, String dstId, String newName)	**void**	Copies mail into a folder or record.

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

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

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

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

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.copyMail("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedMailCopy

Description:

Method	Return values	Description
extendedMailCopy(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolena security, String newName)	**Mail**	Copy mail width with associated data into a folder or record.

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

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

Additional:

- When the category parameter is true the original values of the categories will be copied.
- When the keywords parameter is true the original values of the keywords will be copied.
- When the propertyGroups parameter is true the original values of the metadata groups will be copied.
- When the notes parameter is true the original values of the notes will be copied.
- When the security parameter is true the original value of the security will be copied.
- When newName is set the mail name is renamed.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Mail mail = ws.mail.extendedMailCopy("e3831cc4-30f0-419a-8fbf-a3418472ada
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailChildren

Description:

Method	Return values	Description
getMailChildren(String uuid)	**List<Mail>**	Returns a list of all mails which their parent is fldId.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Mail mail : ws.mail.getMailChildren("ada67d44-b081-4b23-bdc1-74181ca
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValidMail

Description:

Method	Return values	Description
isValidMail(String uuid)	**Boolean**	Returns a boolean that indicates if the node is a mail or not.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            // Return false
            System.out.println(ws.mail.isValidMail("ada67d44-b081-4b23-bdc1-74181cafb

            // Return true
            System.out.println(ws.mail.isValidMail("e3831cc4-30f0-419a-8fbf-a3418472a
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

getMailPath

Description:

Method	Return values	Description
getMailPath(String uuid)	**String**	Converts mail UUID to mail path.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.mail.getMailPath("e3831cc4-30f0-419a-8fbf-a3418472a
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createAttachment

Description:

Method	Return values	Description
createAttachment(String uuid, String docName, InputStream is)	**Document**	Adds an attachement to the mail.

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/logo.png");
            ws.mail.createAttachment("e3831cc4-30f0-419a-8fbf-a3418472ada5", "logo.pn
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteAttachment

Description:

Method	Return values	Description
deleteAttachment(String uuid, String docId)	**void**	Delete a mail attachment.
The value of the parameter **docId** parameter can be a valid document **UUID**.		

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.deleteAttachment("e3831cc4-30f0-419a-8fbf-a3418472ada5", "6dcb02d
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAttachments

Description:

Method	Return values	Description
getAttachments(String uuid)	**List<Document>**	Retrieves a list of all the attachment documents of the mails.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (Document doc : ws.mail.getAttachments("e3831cc4-30f0-419a-8fbf-a3418
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMailWithAttachments

Description:

Method	Return values	Description
sendMailWithAttachments(List<String> to, List<String> cc, List<String> bcc, List<String> replyTo, String subject, String body, List<String> docsId, String uuid)	**void**	Send mail message with attachement.

The values of the uuid parameter should be a folder or record UUID. The uuid parameter indicate where the mail will be stored in the repository after is sent.

Other parameters:

- to are a list of mail accounts destination.

- subject is the mail subject.
- cc, bcc, replyTo is a list of mail accounts destination (optional)
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail (optional).

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            List<String> to = Arrays.asList("gnu.java.sergio@mail.com");
            List<String> cc = new ArrayList<>();
            List<String> bcc = new ArrayList<>();
            List<String> replyTo = new ArrayList<>();
            List<String> docsId = Arrays.asList("46762f90-82c6-4886-8d21-ad3017dd78a7
            ws.mail.sendMailWithAttachments(to, cc, bcc, replyTo, "some subject", "so
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMailWithAttachments

Description:

Method	Return values	Description
sendMailWithAttachments(String from, List<String> to, List<String> cc, List<String> bcc, List<String> replyTo, String subject, String body, List<String> docsId, String uuid)	**void**	Send mail message with attachement.
The values of the uuid parameter should be a folder or record UUID. The uuid parameter indicate where the mail will be stored in the repository after is sent.		

Other parameters:

- from is optional and maybe set with empty value
- to are a list of mail accounts destination.
- subject is the mail subject.
- cc, bcc, replyTo is a list of mail accounts destination (optional)
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail (optional).

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            String from = "some@nomail.com";
            List<String> to = Arrays.asList("gnu.java.sergio@mail.com");
            List<String> cc = new ArrayList<>();
            List<String> bcc = new ArrayList<>();
            List<String> replyTo = new ArrayList<>();
            List<String> docsId = Arrays.asList("46762f90-82c6-4886-8d21-ad3017dd78a7
            ws.mail.sendMailWithAttachments(from, to, cc, bcc, replyTo, "some subject
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importEml

Description:

Method	Return values	Description
importEml(String uuid, String title, InputStream is)	**Mail**	Import a mail in EML format.

The values of the **uuid** parameter should be a folder or record UUID. The **uuid** parameter indicates where the mail will be

stored in the repository after is imported.

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.eml");
            Mail mail = ws.mail.importEml("85f07f05-1641-47e8-891f-0ea30643554e", "so
            System.out.println(mail);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importMsg

Description:

Method	Return values	Description
importMsg(String uuid, String title, InputStream is)	**Mail**	Import a mail in MSG format.

The values of the **uuid** parameter should be a folder or record UUID. The **uuid** parameter indicates where the mail will be stored in the repository after is sent.

Example:

```
package com.openkm;

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

```
import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.msg");
            Mail mail = ws.mail.importMsg("85f07f05-1641-47e8-891f-0ea30643554e", "so
            System.out.println(mail);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setMailTitle

Description:

Method	Return values	Description
setMailTitle(String uuid, String title)	**void**	Sets the mail title.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.setMailTitle("9d5dd110-db99-4d72-8cf7-610b027a4822", "new title")
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMail

Description:

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

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            StringBuffer body = new StringBuffer();
            body.append("Test message body.");
            List<String> recipients = new ArrayList<>();
            recipients.add("some@mail.com");
            ws.mail.sendMail(recipients, "Testing sending mail from OpenKM", body.toS
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMail

Description:

Method	Return values	Description
sendMail(String from, List<String> recipients, String subject, String body)	**void**	Sends a mail.
Other parameters: • from is optional and maybe set with empty value		

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);
            String from = "some@nomail.com";
            StringBuffer body = new StringBuffer();
            body.append("Test message body.");
            List<String> recipients = new ArrayList<>();
            recipients.add("some@mail.com");
            ws.mail.sendMail(to, recipients, "Testing sending mail from OpenKM", body
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setMailDispositionStage

Description:

Method	Return values	Description
setMailDispositionStage(String uuid, long stage)	**void**	Set the disposition stage

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            long stage = 1;
            ws.mail.setMailDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", s
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

setMailDescription

Description:

Method	Return values	Description
setMailDescription(String uuid, String description)	**void**	Set the description.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.mail.setMailDescription("b405a504-d8cb-4166-ac51-22f68acee8c5", "some
            Mail mail = ws.mail.getMailProperties("b405a504-d8cb-4166-ac51-22f68acee8
            System.out.println("Description: " + mail.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailContent

Description:

Method	Return values	Description
getMailContent(String mailId)	**InputStream**	Retrieve mail content - binary data - of the current mail version.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
```

```
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            OutputStream fos = new FileOutputStream("/home/openkm/test.eml");
            InputStream is = ws.mail.getMailContent("b405a504-d8cb-4166-ac51-22f68ace
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizardMail

Description:

Method	Return values	Description
createWizardMail(String uuid, String title, InputStream is, String type)	**WizardNode**	Create a new document with a wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

Available types:

- Mail.ORIGIN_MSG
- Mail.ORIGIN_EML

The WizardNode contains a list of pending actions what should be done to complete the process of document creation. These might be:

- Add keyword
- Add Categories
- Add Metadata

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/sample2.eml");
            WizardNode wn = ws.mail.createWizardMail("4c195453-246b-4ce9-86ba-b84e68d
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailThumbnail

Description:

Method	Return values	Description
getMailThumbnail(String mailId, ThumbnailType type)	**InputStream**	Returns thumbnail image data.

Available types:

- ThumbnailType.THUMBNAIL_PROPERTIES (shown in properties view)
- ThumbnailType.THUMBNAIL_LIGHTBOX (shown in light box)
- ThumbnailType.THUMBNAIL_SEARCH (shown in search view)

Each thumbnail type has its own image dimensions.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

        try {
            ws.login(user, password);
            OutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/thumbnai
            InputStream is = ws.mail.getMailThumbnail("1778f0b5-672c-48c1-b54e-494c18
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailsPaginated

Description:

Method	Return values	Description
getMailsPaginated(String context, int offset, int limit, MailFilterQuery filter, String orderColumn, bool orderAsc)	**NodeList**	Returns a list of emails paginated results filtered by the values of the MailFilterQuery parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.
- The parameter "filter" must be the canonical class name of the class which implements the NodeProperties interface.
- The parameter "orderColumn" can have the following values: uuid, subject, from, sentDate and hasAttachments.

- The parameter "orderAsc" can be *"true"* in case of ascending order and *"false"* otherwise.

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.AttachmentEnum;
import com.openkm.sdk4j.bean.MailFilterQuery;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.bean.NodeList;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            MailFilterQuery filter = new MailFilterQuery();
            filter.setSubject("");
            filter.setAttachments(AttachmentEnum.ALL);
            String orderColumn = "subject";
            NodeList nodes = ws.mail.getMailsPaginated("/okm:root", 0, 20, filter, or

            for (Node node : nodes.getNodes()) {
                System.out.println(node);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailAccounts

Description:

Method	Return values	Description

getMailAccounts()	**List<MailAccount>**	Retrieves a list of user email accounts.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            for (MailAccount mailAccount : ws.mail.getMailAccounts()) {
                System.out.println(mailAccount);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailMessages

Description:

Method	Return values	Description
getMailMessages(long accountId, long start)	**MailServerMessages**	Retrieves a list of messages in the email server for an email account.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ExternalMail;
import com.openkm.sdk4j.bean.MailServerMessages;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

```
        try {
            ws.login(user, password);
            long accoundId = 1;
            long start = 1;
            MailServerMessages msg = ws.mail.getMailMessages(accoundId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                System.out.println(mail);
            }

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

addMailAccount

Description:

Method	Return values	Description
addMailAccount(MailAccount mailAccount)	**void**	Add an email account.

It is a good practice to create an account, verify the mail configuration with the testMailAccount(MailAccount mail).

When you do **not set the user**, the account will be **added by default to the logged user**.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            MailAccount mailAcount = new MailAccount();
            mailAcount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
            mailAcount.setMailUser("test@none.com");
            mailAcount.setMailPassword("123456");
            mailAcount.setMailHost("imap.gmail.com");
            mailAcount.setMailFolder("OpenKM");
            mailAcount.setActive(true);
            mailAcount.setRecursive(true);
            ws.mail.addMailAccount(mailAcount);
```

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

updateMailAccount

Description:

Method	Return values	Description
updateMailAccount(MailAccount mailAccount)	**void**	Update the main configuration data of an email account.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            MailAccount mailAcount = new MailAccount();
            mailAcount.setId(2);
            mailAcount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
            mailAcount.setMailUser("testupdate@none.com");
            mailAcount.setMailPassword("123456789");
            mailAcount.setMailHost("imap.gmail.com");
            mailAcount.setMailFolder("OpenKM");
            mailAcount.setActive(true);
            mailAcount.setRecursive(true);
            ws.mail.updateMailAccount(mailAcount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

testMailAccount

Description:

Method	Return values	Description

testMailAccount(MailAccount mailAccount)	**void**	Check email account connection.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            MailAccount mailAcount = new MailAccount();
            mailAcount.setMailProtocol(MailAccount.PROTOCOL_IMAPS);
            mailAcount.setMailUser("testupdate@none.com");
            mailAcount.setMailPassword("123456789");
            mailAcount.setMailHost("imap.gmail.com");
            mailAcount.setMailFolder("OpenKM");
            mailAcount.setActive(true);
            mailAcount.setRecursive(true);

            // Test mail account
            ws.mail.testMailAccount(mailAcount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteMailAccount

Description:

Method	Return values	Description
deleteMailAccount(long mailAccountId)	**void**	Delete email account.

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
```

```
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long mailAccountId = 2;
            ws.mail.deleteMailAccount(mailAccountId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importMailMessages

Description:

Method	Return values	Description
importMailMessages(long mailAccountId, List<Long> messageIds)	**void**	Import email account messages.

Example:

```
package com.openkm;

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

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ExternalMail;
import com.openkm.sdk4j.bean.MailServerMessages;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            List<Long> messageIds = new ArrayList<>();

            long mailAccountId = 1; // Valid mailAccountId
            long start = 1;
            MailServerMessages msg = ws.getMailMessages(mailAccountId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                messageIds.add(mail.getUid());
            }

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

```
}
```

createMailFilter

Description:

Method	Return values	Description
createMailFilter(long mailAccountId, MailFilter mailFilter)	**void**	Add email account filter.

Example

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            long mailAccountId = 2; // Valid mailAccountId

            MailFilter filter = new MailFilter();
            filter.setOrder(0);
            filter.setGrouping(false);
            filter.setExclusive(true);
            filter.setActive(false);
            filter.setNode("a9d5c158-c7b1-4771-b9c3-b8d24f5a2645");

            ws.mail.createMailFilter(mailAccountId, filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateMailFilter

Description:

Method	Return values	Description
updateMailFilter(MailFilter mailFilter)	**void**	Update email account filter.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            MailFilter filter = new MailFilter();
            filter.setId(2);  // Valid filter id
            filter.setOrder(0);
            filter.setGrouping(false);
            filter.setExclusive(true);
            filter.setActive(true);
            filter.setNode("a9d5c158-c7b1-4771-b9c3-b8d24f5a2645");

            ws.mail.updateMailFilter(filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteMailFilter

Description:

Method	Return values	Description
deleteMailFilter(long mailFilterId)	**void**	Delete email account filter.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            long filterId = 1; // Valid filter id
            ws.mail.deleteMailFilter(filterId);
```

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

createMailRule

Description:

Method	Return values	Description
createMailRule(long filterId, MailFilterRule rule)	**void**	Create email filter rule.

Mail account filter parameters:

- **Field**: Sets the mail field that will be evaluated by the rule.

  Available options are:

  - MailFilterRule.FIELD_FROM
  - MailFilterRule.FIELD_TO
  - MailFilterRule.FIELD_SUBJECT
  - MailFilterRule.FIELD_CONTENT
  - MailFilterRule.FIELD_ATTACHMENT

- **Operation:** Set the operation that will be done for the evaluation process.

  Available options are:

  - MailFilterRule.OPERATION_EQUALS
  - MailFilterRule.OPERATION_NOT_EQUALS
  - MailFilterRule.OPERATION_CONTAINS
  - MailFilterRule.OPERATION_ENDS_WITH
  - MailFilterRule.OPERATION_STARTS_WITH

- ***Value***: *Sets the value that will be used for the evaluation process.*
- ***Active:*** *Sets rule enabled or not.*

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            long filterId = 2;// Valid filter id

            MailFilterRule rule = new MailFilterRule();
            rule.setOperation(MailFilterRule.OPERATION_CONTAINS);
            rule.setValue("test");
            rule.setField(MailFilterRule.FIELD_FROM);
            rule.setActive(false);

            ws.mail.createMailRule(filterId, rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateMailRule

Description:

Method	Return values	Description
updateMailRule(MailFilterRule rule)	**void**	Update email coount rule.

Mail account filter parameters:

- **Field**: Sets the mail field that will be evaluated by the rule.

Available options are:

- MailFilterRule.FIELD_FROM
- MailFilterRule.FIELD_TO
- MailFilterRule.FIELD_SUBJECT
- MailFilterRule.FIELD_CONTENT
- MailFilterRule.FIELD_ATTACHMENT

- **Operation:** Set the operation that will be done for the evaluation process.

  Available options are:

  - MailFilterRule.OPERATION_EQUALS
  - MailFilterRule.OPERATION_NOT_EQUALS
  - MailFilterRule.OPERATION_CONTAINS
  - MailFilterRule.OPERATION_ENDS_WITH
  - MailFilterRule.OPERATION_STARTS_WITH

- ***Value**: Sets the value that will be used for the evaluation process.*
- ***Active:** Sets rule enabled or not.*

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            MailFilterRule rule = new MailFilterRule();
            rule.setId(2);
            rule.setOperation(MailFilterRule.OPERATION_EQUALS);
            rule.setValue("test");
            rule.setField(MailFilterRule.FIELD_FROM);
            rule.setActive(false);

            ws.mail.updateMailRule(rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteMailRule

Description:

Method	Return values	Description

deleteMailRule(long ruleId)	**void**	Delete a rule of an email account.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            long ruleId = 1; // Valid rule id
            ws.mail.deleteMailRule(ruleId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailFilterRules

Description:

Method	Return values	Description
getMailFilterRules(long filterId)	**List<MailFilterRule>**	Retrieve a list rules by filter.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            long filterId = 2; // Valid filter id
```

```
            for (MailFilterRule mailFilterRule : ws.mail.getMailFilterRules(filterId)
                System.out.println(mailFilterRule);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forwardEmail

Description:

Method	Return values	Description
forwardEmail(String uuid, List<String> users, List<String> roles, List<String> mails, String message)	**void**	Forward a email to a list of users, roles or external emails.

Example:

```
package com.openkm;

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

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

public class Test {

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

        try {
            ws.login(user, password);

            String mailUuid = "f123a950-0329-4d62-8328-0ff500fd42db";
            List<String> users = new ArrayList<>();
            users.add("userTest");
            List<String> roles = new ArrayList<>();
            roles.add("ROLE_USER");
            List<String> mails = new ArrayList<>();
            mails.add("test@none.com");

            ws.mail.forwardEmail(mailUuid, users, roles, mails, "Any message");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPdf

Description:

Method	Return values	Description
getPdf(String uuid)	**InputStream**	Returns a PDF of the email.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.apache.commons.io.IOUtils;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class Test {

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

        try {
            ws.login(user, password);
            Mail mail = ws.mail.getMailProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e
            InputStream is = ws.mail.getPdf(mail.getUuid());
            OutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/" + mail
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveMailAsPdf

Description:

Method	Return values	Description
saveMailAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)	**Document**	Save the mail as PDF in the OpenKM repository.
The value of the **uuid** parameter should be a Mail UUID. When the parameter newName value is null, the document will preserve the same name.		

Additional:

- When the propertyGroups parameter is true the metadata groups of the source are copied to the target.
- When the security parameter is true the security of the source is copied to the target.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.bean.Mail;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            Mail mail = ws.mail.getMailProperties("7f6c5c0b-a66b-4782-9595-e14b402a18
            Document docPdf = ws.mail.saveMailAsPdf(mail.getUuid(), mail.getSubject()
            System.out.println("Document pdf: " + docPdf.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Node samples

Basics

Example of uuid:

- Using UUID -> "**373bcdd0-c082-4e7b-addd-e10ef813946e**";

Suggested code sample

First, you must create the webservice object:

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

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

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

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Node methods from "**node**" class as is shown below:

```
ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb")
```

Methods

getNodeByUuid

Description:

Method	Return values	Description
getNodeByUuid(String uuid)	**Node**	Get a node by uuid.

Example:

```
package com.openkm;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            Node node = ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb")
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistory

Description:

Method	Return values	Description
getVersionHistory(String uuid)	**List<Version>**	Returns a list of the version history of a document.

Example:

```
package com.openkm;

import java.util.List;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<Version> versions = ws.node.getVersionHistory("3767deb4-21e7-4272-82
            for (Version version : versions) {
                System.out.println(version);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

restoreVersion

Description:

Method	Return values	Description
restoreVersion(String uuid, String versionName)	**void**	Restore a document to a specific version.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.restoreVersion("3767deb4-21e7-4272-82be-fece5384fbab", "1.2");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteVersion

Description:

Method	Return values	Description
deleteVersion(String uuid, String versionName)	**void**	Delete a specific version.

Example:

```
package com.openkm;

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

public class Test {

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

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

```
            ws.node.deleteVersion("5aab162d-df4f-4392-96c9-8fc1607e3903", "1.1");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeVersionHistory

Description:

Method	Return values	Description
purgeVersionHistory(String uuid)	**void**	Purge version history of a document.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.purgeVersionHistory("3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

mayBePromotedAsRecord

Description:

Method	Return values	Description
mayBePromotedAsRecord(String uuid, boolean fullEvaluation)	**PromoteAsRecordEvaluation**	Returns a PromoteAsRecordEvaluation object.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            PromoteAsRecordEvaluation pre = ws.node.mayBePromotedAsRecord("3767deb4-2
            System.out.println(pre);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

promoteAsRecord

Description:

Method	Return values	Description
promoteAsRecord(String uuid)	**void**	Promote as record.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.promoteAsRecord("3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

degradeRecord

Description:

Method	Return values	Description
degradeRecord(String uuid)	**void**	Degrade a record.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.degradeRecord("3767deb4-21e7-4272-82be-fece5384fbab");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isElectronicRecordPath

Description:

Method	Return values	Description
isElectronicRecordPath(String uuid)	**boolean**	Returns true when the node is into an electronic record.

Return true when one of the parents of the node is an electronic record.

Example:

```
package com.openkm;

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

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.node.isElectronicRecordPath("3767deb4-21e7-4272-82b
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getElectronicRecordInPath

Description:

Method	Return values	Description
getElectronicRecordInPath(String uuid)	**Record**	Get the electronic record in the path.

Returns the first electronic record in the path of the node.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Record record = ws.node.getElectronicRecordInPath("3767deb4-21e7-4272-82b
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenNodesPaginated

Description:

Method	Return values	Description
getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes)	**SimpleNodeBaseList**	Get children nodes paginated.

Available filteredTypes values:

```
SimpleNodeBase.TYPE_FOLDER
SimpleNodeBase.TYPE_DOCUMENT
SimpleNodeBase.TYPE_MAIL
SimpleNodeBase.TYPE_RECORD
```

You should do:

```
List<Integer> filteredTypes = new ArrayList<>();
filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
```

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
```

```
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
            filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesPaginated(uuid, 0,
                    filteredTypes);
            System.out.println("Filtered Elements: " + listNode.getFilteredElements())
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenNodesPaginated

Description:

Method	Return values	Description
getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes, String pluginName)	**SimpleNodeBaseList**	Get children nodes paginated.

Available filteredTypes values:

```
SimpleNodeBase.TYPE_FOLDER
SimpleNodeBase.TYPE_DOCUMENT
SimpleNodeBase.TYPE_MAIL
SimpleNodeBase.TYPE_RECORD
```

You should do:

```
List<Integer> filteredTypes = new ArrayList<>();
filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
```

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

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

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

- limit=10
- offset=0

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

- limit=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(SimpleNodeBase.TYPE_FOLDER);
            filteredTypes.add(SimpleNodeBase.TYPE_DOCUMENT);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesPaginated(uuid, 0,
                    filteredTypes, "");
            System.out.println("Filtered Elements: " + listNode.getFilteredElements());
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
```

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

getChildrenNodesByCategoryPaginated

Description:

Method	Return values	Description
getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes)	**SimpleNodeBaseList**	Get children nodes by category paginated.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;
```

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

public class Test {

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

        try {
            ws.login(user, password);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(1);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesByCategoryPaginated
                    filteredTypes);
            System.out.println("Filtered Elements: " + listNode.getFilteredElements()
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenNodesByCategoryPaginated

Description:

Method	Return values	Description
getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, boolean orderAsc, List<Integer> filteredTypes, String pluginName)	**SimpleNodeBaseList**	Get children nodes by category paginated.

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

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

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

- limit=10

- offset=0

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

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodesPaginationInfo;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseList;
import com.openkm.sdk4j.impl.OKMWebservices;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<Integer> filteredTypes = new ArrayList<>();
            filteredTypes.add(1);
            // Folder UUID or Record UUID
            String uuid = "39479efe-de5e-468e-91a7-24d2aa3f8837";
            SimpleNodeBaseList listNode = ws.node.getChildrenNodesByCategoryPaginated
                    filteredTypes, "");
            System.out.println("Filtered Elements: " + listNode.getFilteredElements()
            System.out.println("Total Elements: " + listNode.getTotalElements());
            for (SimpleNodeBase simpleNodeBase : listNode.getNodes()) {
                System.out.println(simpleNodeBase.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getBreadcrumb

Description:

Method	Return values	Description
getBreadcrumb(String uuid)	**List<BreadCrumbItem>**	Get breadcrumb.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            List<BreadCrumbItem> list = ws.node.getBreadcrumb("39479efe-de5e-468e-91a
            for (BreadCrumbItem item : list) {
                System.out.println(item.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

subscribe

Description:

Method	Return values	Description
subscribe(String uuid)	**void**	Adds a subscription to a node.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.subscribe("39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

unsubscribe

Description:

Method	Return values	Description
unsubscribe(String uuid)	**void**	Delete a subscription to a node.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.unsubscribe("39479efe-de5e-468e-91a7-24d2aa3f8837");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importZip

Description:

Method	Return values	Description
importZip(String uuid, InputStream is)	**void**	Import a zip file.

Example:

```
package com.openkm;

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

import org.apache.commons.io.IOUtils;

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

```
public class Test {

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

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/import.zip"
            ws.node.importZip("212e7c1f-443d-4aac-a12c-0b818ca03419", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unZip

Description:

Method	Return values	Description
unZip(String uuid, String dstId)	**void**	Unzip file.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            String uuid = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801"; // zip File
            String dstId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7"; // destination uui
            ws.node.unZip(uuid, dstId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

exportZip

Description:

Method	Return values	Description
exportZip(List<String> uuids, boolean withPath, boolean background)	**InputStream**	Export as a zip file.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;

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

public class Test {

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

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

            OutputStream fos = new FileOutputStream("/home/openkm/export.zip");

            InputStream is = ws.node.exportZip(uuids, true, true);
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodesFiltered

Description:

Method	Return values	Description
getNodesFiltered(List<String> uuids)	**List<Node>**	Return a list of nodes.

Example:

```
package com.openkm;
```

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

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

public class Test {

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

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

            List<Node> nodes = ws.node.getNodesFiltered(uuids);
            for (Node node : nodes) {
                System.out.println(node);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

evaluateDownloadZip

Description:

Method	Return values	Description
evaluateDownloadZip(List<String> uuids)	**ZipDownloadEvaluationResult**	Return a ZipDownloadEvaluationResult object.

Example:

```
package com.openkm;

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

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

public class Test {

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

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

            ZipDownloadEvaluationResult result = ws.node.evaluateDownloadZip(uuids);
            System.out.println(result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

generateDownloadToken

Description:

Method	Return values	Description
generateDownloadToken(String uuid, boolean preview)	**String**	Generate a node download link.

When parameter preview is set to true, the token will be generated for previewing purpose.

The preview token expires by default in a minute.

The user must build the download URL with the returned token:

- Download: http://localhost:8080/openkm/Download?DTK=token
- Preview: http://localhost:8080/openkm/Download?PTK=token

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            System.out.println(ws.node.generateDownloadToken("b2f88679-e3fd-4f97-bf0e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

restore

Description:

Method	Return values	Description
restore(String uuid)	**Node**	Restore a node

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            Node node = ws.node.restore("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            System.out.println(node);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasNodesLockedByOtherUser

Description:

Method	Return values	Description
hasNodesLockedByOtherUser(String uuid)	**booelan**	If the node is blocked by other users

Example:

```
package com.openkm;

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

public class Test {
```

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

        try {
            ws.login(user, password);
            if(ws.node.hasNodesLockedByOtherUser("0f6463f3-4d36-4091-b518-4fe7c353ee7
             System.out.println("Is blocked");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setComment

Description:

Method	Return values	Description
setComment(String uuid, String versionName, String comment)	**void**	Sets the comment for a specific node version.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);
            ws.node.setComment("1ec49da9-1746-4875-ae32-9281d7303a62", "1.14", "Updat
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPaginatorConfig

Description:

Method	Return values	Description

getPaginatorConfig(String pluginName)	**NodePaginatorConfig**	Return a NodePaginatorConfig object.

Example:

```
package com.openkm;

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

public class Test {

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

        try {
            ws.login(user, password);

            NodePaginatorConfig nodePaginatorConfig = ws.node.getPaginatorConfig("com
            System.out.println(nodePaginatorConfig);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPaginatorPlugins

Description:

Method	Return values	Description
getPaginatorPlugins()	**PaginatorPluginList**	Return a PaginatorPluginList object.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PaginatorPlugin;
import com.openkm.sdk4j.bean.PaginatorPluginList;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

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

        try {
            ws.login(user, password);
            PaginatorPluginList list = ws.node.getPaginatorPlugins();
```

```
            for (PaginatorPlugin paginatorPlugin : list.getPaginatorPlugins()) {
                System.out.println("Name: " + paginatorPlugin.getName());
                System.out.println("ObjClass: " + paginatorPlugin.getObjClass());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Note samples

Basics

Example of uuid:

- Using UUID -> "**373bcdd0-c082-4e7b-addd-e10ef813946e**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Note methods from " **note**" class as is shown below:

```
ws.note.addNote("373bcdd0-c082-4e7b-addd-e10ef813946e", "the note text");
```

Methods

addNote

Description:

Method	Return values	Description
addNote(String uuid, String text)	**Note**	Adds a note to a node and returns an object Note.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.note.addNote("373bcdd0-c082-4e7b-addd-e10ef813946e", "the note text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNote

Description:

Method	Return values	Description
getNote(String noteId)	**Note**	Retrieves the note.

The noteId is an UUID.

The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("373bcdd0-c082-4e7b-addd-e10ef813946
            if (notes.size() > 0) {
                System.out.println(ws.getNote(notes.get(0).getPath()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

deleteNote

Description:

Method	Return values	Description
deleteNote(String noteId)	**Note**	Delete a note.

> The noteId is an UUID.
>
> The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("373bcdd0-c082-4e7b-addd-e10ef813946
            if (notes.size() > 0) {
                ws.note.deleteNote(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setNote

Description:

Method	Return values	Description
setNote(String noteId, String text)	**void**	Changes the note text.

The noteId is an UUID.

The object Node has a variable named path, in that case the path contains an UUID.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("373bcdd0-c082-4e7b-addd-e10ef813946
            if (notes.size() > 0) {
                ws.note.setNote(notes.get(0).getPath(), "text modified");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

listNotes

Description:

Method	Return values	Description
listNotes(String uuid)	**List<Note>**	Retrieves a list of all the notes of a node.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Note;

public class Test {

    public static void main(String[] args) {
```

```
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("373bcdd0-c082-4e7b-addd-e10ef813946
            for (Note note : notes) {
                System.out.println(note);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotesHistory

Description:

Method	Return values	Description
getNotesHistory(String uuid)	**List<NoteHistory>**	Return the historical notes of a node.

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservices;
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NoteHistory;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<NoteHistory> list = ws.note.getNotesHistory("3c68b3a1-c65c-4b1e-84b5
            for (NoteHistory noteHistory : list) {
                System.out.println(noteHistory);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Notification samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Notification methods from " **notification**" class as is shown below:

```
ws.notification.notify(uuids, users, roles, mails, message, false);
```

Methods

notify

Description:

Method	Return values	Description
notify(List<String> uuids, List<String> users, List<String> roles, List<String> mails, String message, boolean attachment)	**void**	Send a mail notification.

The parameter uuids are the UUID of the node (document, folder, mail or record).

The parameter users are a set of OpenKM users to be notified.

The parameter roles are a set of OpenKM roles to be notified.

The parameter mails are a set of email addresses - usually external mails - to be notified.

The parameter message is the content body of the mail.

When attachment value is true the node is attached into the mail.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<String> uuids = new ArrayList<>();
            uuids.add("b153c4b7-3d1c-4589-bd42-0ed0f34fd338");

            List<String> users = new ArrayList<>();
            users.add("test");
            users.add("sochoa");

            List<String> roles = new ArrayList<>();
            roles.add("ROLE_TEST");

            List<String> mails = new ArrayList<>();
            String message = "Body of the message";
            ws.notification.notify(uuids, users, roles, mails, message, false);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

PDF samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is kept in the webservice Object. Then you can use the other API method

At this point you can use all the Pdf methods from "**pdf**" class as is shown below:

```
InputStream is = ws.pdf.getImage("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", 1);
```

Methods

getImage

Description:

Method	Return values	Description
getImage(String uuid, int page, String size)	**InputStream**	Returns the image of a page.

The document must be a PDF or convertible to PDF.

The default value of size parameter is "150". The value is in pixels.

The value of the **uuid** is the UUID of the document.

The **page** is the number of the page in the document starting from the value "1".

The **size** value is optional. When set is used to resize the image of the page, otherwise is used the default value.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            InputStream is = ws.pdf.getImage("e7d6860a-7e0b-4823-bd3d-75f88be1eedf",
            OutputStream fos = new FileOutputStream("/home/openkm/page-1.png");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

split

Description:

Method	Return values	Description
split(String uuid, String dstId, String baseName, List<Integer> pages)	**boolean**	Return true when the pages of the document have been split.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of **baseName** is the base name to create a split document, for example, if baseName value is "test" and pages value is "1,3", the files created will be "test-001.pdf" and "test-003.pdf".

The value of **pages** is the number of pages to be split.

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            boolean value = ws.pdf.split("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be
            if (value) {
                System.out.println("split done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extract

Description:

Method	Return values	Description
extract(String uuid, String dstId, String name, List<Integer> pages)	**boolean**	Return true when the chosen pages of the document have been extracted.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be extracted.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            //extract
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac
                System.out.println("extract done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

remove

Description:

Method	Return values	Description
remove(String uuid, String dstId, String name, List<Integer> pages)	**boolean**	Return true when the pages of the document have been removed.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be removed.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            // remove
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac
                System.out.println("remove done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rotate

Description:

Method	Return values	Description
rotate(String uuid, String dstId, String name, String angle, List<Integer> pages)	**boolean**	Return true when the pages of the document have been rotated.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **angle** is rotation applied in the chosen pages..

The value of **pages** is the number of pages to be rotated.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            // rotate
            List<Integer> pages = new ArrayList<>();
            pages.add(2);
            pages.add(3);
            if (ws.pdf.rotate("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-
                    pages)) {
                System.out.println("rotation done");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

insertPages

Description:

Method	Return values	Description
insertPages(String uuid, String srcId, int sourceFromPage, int sourceToPage, int insertFromPage)	**boolean**	Returns true when the pages have been inserted into the document.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the target document where the pages will be inserted.

The value of **srcId** is the UUID of the source document from which the pages will be extracted.

The value of **sourceFromPage** is the starting page number in the source document for page extraction.

The value of **sourceToPage** is the ending page number in the source document for page extraction. Extraction is inclusive of this page.

The value of **insertFromPage** is the page number in the target document where the extracted pages will be inserted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class PdfExample {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);
            ws.pdf.insertPages("6a54d98b-6454-4484-8172-8359a2b4f2f2", "a573da75-ba1e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Plugin samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Plugin methods from "**plugin**" class as is shown below:

```
ws.plugin.executePluginPost("com.openkm.plugin.rest.TestRestPlugin", parameters, Stri
```

Methods

executePluginPost

Description:

Method	Return values	Description
executePluginPost(String className, Map<String, String> parameters, Class<?> clazz, InputStream is)	**Object**	Return the Object value of execution of a class which implements RestPlugin.

- The "className" value must be the canonical class name of a class which implements RestPlugin interface.
- The "parameters" map are values what will be used by class set in className.
- The "clazz" value should be the Class class of the returned object. It's used by REST for unmarshalling the result of the query.
- The "is" stream allow to uploading a document.

This method execute a REST call at POST.

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> parameters = new HashMap<>();
            parameters.put("param1", "value1");
            parameters.put("param2", "value2");
            String value = (String) ws.plugin.executePluginPost("com.openkm.plugin.re
            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executePostPluginReturnFile

Description:

Method	Return values	
executePluginPostReturnFile(String className, Map<String, String> parameters, InputStream is)	**InputStream**	Return an InputStre implements RestPl

- The "className" value must be the canonical class name of a class which implements RestPlugin interface
- The "parameters" map are values what will be used by class set in className.
- The "is" stream allow to uploading a document.

This method execute a REST call at POST.

The RestPlugin must return a Response object. Take a look at the next sample implementation of the RestPlugin:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlu

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestl

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, fa

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filenam
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(i
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, H
    }
}
```

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            InputStream is = ws.plugin.executePluginPostReturnFile("com.openkm.plugin
            OutputStream fos = new FileOutputStream("/home/user/Desktop/test.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executePluginGet

Description:

Method
executePluginGet(String className, Map<String, String> parameters, Class<?> clazz)

- The "className" value must be the canonical class name of a class which implements RestPlugin interface
- The "parameters" map are values what will be used by class set in className.
- The "clazz" value should be the Class class of the returned object. It's used by REST for unmarshalling the

This method execute a REST call at GET.

You can also execute directly from the browser, take the next url as a sample:

```
http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?className
```

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Browser URL
            // http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?cla
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            parameters.put("docId2", "/okm:root/invoices/00000001_001_of_001.pdf2");
            String value = (String) ws.plugin.executePluginGet("com.openkm.plugin.res
            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executePluginAtGetReturnFile

Description:

Method
executeGetPluginReturnFile(String className, Map<String, String> parameters)

- The "className" value must be the canonical class name of a class which implements RestPlugin interface
- The "parameters" map are values what will be used by class set in className.

This method execute a REST call at GET.

The RestPlugin must return a Response object. Take a look at the next sample implementation of the RestPlugin:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlu

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestl

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, fa

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filenam
        }

        responseHeaders.setContentLength(doc.getActualVersion().getSize());
        InputStreamResource inputStreamResource = new InputStreamResource(i
        log.debug("TestGetDocumentRestPlugin: [BINARY]");
        return new ResponseEntity<>(inputStreamResource, responseHeaders, H
    }
}
```

You can also execute directly from the browser, take the next url as a sample:

```
http://localhost:8080/OpenKM/services/rest/plugin/executeGetPluginReturnFil
```

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Browser URL
            // http://localhost:8180/OpenKM/services/rest/plugin/executeGetPluginRetu
            // http://localhost:8180/OpenKM/services/rest/plugin/executeGetPluginRetu
            Map<String, String> parameters = new HashMap<>();
            parameters.put("docId", "/okm:root/invoices/00000001_001_of_001.pdf");
            InputStream is = ws.plugin.executePluginGetReturnFile("com.openkm.plugin.
            OutputStream fos = new FileOutputStream("/home/jllort/Escritorio/test.pdf
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Property samples

Basics

The value of this parameter can be a valid document, folder, mail or record **UUID**.

Example of nodeId:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Repository methods from "**repository**" class as is shown below:

```
ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-b
```

Methods

addCategory

Description:

Method	Return values	Description
addCategory(String uuid, String catId)	**void**	Set a relation between a category and a node.
The value of the catId parameter should be a category folder UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeCategory

Description:

Method	Return values	Description
removeCategory(String uuid, String catId)	**void**	Removes a relation between a category and a node.
The value of the catId parameter should be a category folder UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.removeCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addKeyword

Description:

Method	Return values	Description
addKeyword(String uuid, String keyword)	**void**	Add a keyword in a node.

The keyword should be a single word without spaces, formats allowed:

- "test"
- "two_words" (the character "_" is used for the junction).

We also we suggest you to add keyword in lowercase format, because OpenKM is case sensitive.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.addKeyword("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeKeyword

Description:

Method	Return values	Description
removeKeyword(String uuid, String keyword)	**void**	Removes a keyword from a node.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.removeKeyword("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "test")
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setEncryption

Description:

Method	Return values	Description
setEncryption(String uuid, String cipherName)	**void**	Marks a document as an en cripted binary data in the repository.

The parameter nodeId should be a document node.

The parameter cipherName saves information about the encription mechanism.

This method does not perform any kind of encryption, simply mark in the database that a document is encrypted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
```

```
            ws.login(user, password);
            ws.property.setEncryption("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "phrase
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unsetEncryption

Description:

Method	Return values	Description
unsetEncryption(String uuid)	**void**	Marks a document is a normal binary data into repository.

The parameter **uuid** should be a document node.

This method does not perform any kind of encryption, simply mark into the database that a document has been uncrypted.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.unsetEncryption("eee9d70f-6af9-4783-82a7-b8ac94c234b6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setSigned

Description:

Method	Return values	Description

setSigned(String uuid, boolean signed)	**void**	Marks a document as signed or unsigned binary data into the repository.

The parameter **uuid** should be a document node.

This method does not perform any kind of digital signature process, simply mark into the database that a document is signed.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.property.setSigned("eee9d70f-6af9-4783-82a7-b8ac94c234b6", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isSigned

Description:

Method	Return values	Description
isSigned(String uuid)	**boolean**	Returns a boolean that indicates if the document is signed or not.

The parameter **uuid** should be a document node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            System.out.println("Is the document signed: " + ws.property.isSigned("633
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

PropertyGroup samples

Basics

From the older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".

Although we understand this name does not help a lot to identify these methods with metadata ones, for historical reason, we continue maintaining the nomenclature.

For more information about Metadata.

The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields. The metadata field of type date values is stored in the application in ISO-8601 basic format.

To convert the retrieved metadata field of type date to a valid date use:

```
Calendar cal = ISO8601.parseBasic(metadataFieldValue);
```

To save the date value in the metadata field of type date use:

```
Calendar cal = Calendar.getInstance(); // Present date
String metadataFieldValue = ISO8601.formatBasic(cal);
// metadataFieldValue can be saved into repository metadata field of type dat
```

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API methods.

At this point you can use all the Property Group methods from "**propertyGroup**" class as is shown below:

```
ws.propertyGroup.addPropertyGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consul
```

Methods

addPropertyGroup

Description:

Method	Return values	Descr
addPropertyGroup(String uuid, String grpName, Map<String, String> propertiesMap)	**void**	Adds an empty me node.

The grpName should be a valid Metadata group name.

It is not mandatory to set in the propertiesMap parameter all fields values, is enough with the fields you wish to cha

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Date" name="okp:technology.date" type="date" />
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <checkbox label="Important" name="okp:technology.important"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
  <option label="Alfa" value="t1"/>
  <option label="Beto" value="t2"/>
  <option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

Where \"**t1**\" and \"**t2**\" are valid values and character "**,**" is used as separator.

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:technology.type", "[ \"t1\", \"t2\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.language", "[ \"java\" ]");
            properties.put("okp:technology.important", String.valueOf(true));
            ws.propertyGroup.addPropertyGroup("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5",
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removePropertyGroup

Description:

Method	Return values	Description

removePropertyGroup(String uuid, String grpName)	**void**	Removes a metadata group of a node.
The grpName should be a valid Metadata group name.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.propertyGroup.removePropertyGroup("8cd1e072-8595-4dd3-b121-41d622c43f0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroups

Description:

Method	Return values	Description
getPropertyGroups(String uuid)	**List<PropertyGroup>**	Retrieves a list of metadata groups assigned to a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
```

```
            for (PropertyGroup pGroup : ws.propertyGroup.getPropertyGroups("8cd1e072-
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllPropertyGroups

Description:

Method	Return values	Description
getAllPropertyGroups()	**List<PropertyGroup>**	Retrieves a list of all metadata groups set into the application.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (PropertyGroup pGroup : ws.propertyGroup.getAllPropertyGroups()) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm(String grpName)	**List<FormElement>**	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupForm("okg:co
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupFormDefinition

Description:

Method	Return values	Description
getPropertyGroupFormDefinition(String grpName)	**List<FormElement>**	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupFormDefiniti
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupFormDefinition

Description:

Method	Return values	Description
getPropertyGroupFormDefinition(String grpName, String uuid)	**List<FormElement>**	Retrieves a list of all metadata group elements definition.

The grpName should be a valid Metadata group name.

The method is usually used to display empty form elements for creating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
```

```
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupFormDefiniti
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupForm

Description:

Method	Return values	Description
getPropertyGroupForm(String uuid, String grpName)	**List<FormElement>**	Retrieves a list of metadata values of a node.
The grpName should be a valid Metadata group name.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupForm("8cd1e0
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setPropertyGroupProperties

Description:

	Return

Method	values	Descri
setPropertyGroupProperties(String uuid, String grpName, Map<String, String> properties)	**void**	Changes the metadat node.

The grpName should be a valid Metadata group name.

Before changing metadata, you **should have the group added in the node** (see addGroup method) otherwise will error.

It is not mandatory to set in the properties parameterMap all fields values, is enough with the fields you wish to cha

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Date" name="okp:technology.date" type="date" />
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <checkbox label="Important" name="okp:technology.important"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type multiple like this:

```
<select label="Type" name="okp:technology.type" type="multiple">
  <option label="Alfa" value="t1"/>
  <option label="Beto" value="t2"/>
  <option label="Omega" value="t3" />
</select>
```

You should do:

```
properties.put("okp:technology.type", "[\"t1\", \"t2\"]");
```

Where \"**one**\" and \"**two**\" are valid values and character "**,**" is used as separator.

Another option:

```
Gson gson = new Gson();
List<String> values = new ArrayList<>();
values.add("t1");
values.add("t2");
properties.put("okp:technology.type", gson.toJson(values));
```

Example:

```
package com.openkm;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> properties = new HashMap<>();
            properties.put("okp:technology.type", "[ \"t1\", \"t2\" ]");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:technology.date", ISO8601.formatBasic(cal));
            properties.put("okp:technology.comment", "comment sample");
            properties.put("okp:technology.description", "description sample");
            properties.put("okp:technology.language", "[ \"java\" ]");
            properties.put("okp:technology.important", String.valueOf(true));
            ws.propertyGroup.setPropertyGroupProperties("8cd1e072-8595-4dd3-b121-41d6
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasPropertyGroup

Description:

Method	Return values	Description
hasPropertyGroup(String uuid, String grpName)	**Boolean**	Returns a boolean that indicate if the node has or not a metadata group.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("Have metadata group:" + ws.propertyGroup.hasPropertyG
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRegisteredPropertyGroupDefinition

Description:

Method	Return values	Description
getRegisteredPropertyGroupDefinition()	**String**	Returns the XML Metada groups definition.

The method only can be executed by super user granted (ROLE_ADMIN member user).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.propertyGroup.getRegisteredPropertyGroupDefinition(
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

registerPropertyGroupDefinition

Description:

Method	Return values	Description
registerPropertyGroupDefinition(InputStream is, String name)	**void**	Set the XML Metada groups definition into the repository.

The method only can be executed by super user granted (ROLE_ADMIN member user).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/PropertyGroups.xml");
            ws.propertyGroup.registerPropertyGroupDefinition(is, "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupSuggestions

Description:

Method	Return values	Descripti

getPropertyGroupSuggestions(String uuid, String grpName, String propName)	**List<String>**	Retrieves a list of a suggest values.

The propName parameter should be a Metadata Select field type.

More information at Creating your own Suggestion Analyzer and Metadata Select field.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-
<property-groups>
  <property-group label="Technology" name="okg:technology">
    <select label="Type" name="okp:technology.type" type="multiple">
      <option label="Alfa" value="t1"/>
      <option label="Beta" value="t2" />
      <option label="Omega" value="t3" />
    </select>
    <select label="Language" name="okp:technology.language" type="simple">
      <option label="Java" value="java"/>
      <option label="Python" value="python"/>
      <option label="PHP" value="php" />
    </select>
    <input label="Comment" name="okp:technology.comment"/>
    <textarea label="Description" name="okp:technology.description"/>
    <input label="Link" type="link" name="okp:technology.link"/>
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (String value : ws.propertyGroup.getPropertyGroupSuggestions("8cd1e07
                System.out.println(value);
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupProperties

Description:

Method	Return values	Description
getPropertyGroupProperties(String uuid, String grpName)	**Map<String, String>**	Retrieves a map - (key,value) pairs - with Metada group values of a node.

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> properties = new HashMap<>();
            properties = ws.propertyGroup.getPropertyGroupProperties("8cd1e072-8595-4

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupsByVersion

Description:

Method	Return values	Description
getPropertyGroupsByVersion(String uuid, String versionName)	**List<PropertyGroup>**	Retrieves a list of metadata groups assigned to a specific version of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.PropertyGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (PropertyGroup pGroup : ws.propertyGroup.getPropertyGroupsByVersion("
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupPropertiesByVersion

Description:

Method	Return values	Description
getPropertyGroupPropertiesByVersion(String uuid, String grpName, String versionName)	**Map<String, String>**	Retrieves a map - (key,value) pairs - with Metadata group values assigned to a specific version of a node.

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

```
        try {
            ws.login(user, password);
            Map<String, String> properties = new HashMap<>();
            properties = ws.propertyGroup.getPropertyGroupPropertiesByVersion("118cb0

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertyGroupByVersionForm

Description:

Method	Return values	Description
getPropertyGroupByVersionForm(String uuid, String grpName, String versionName)	**List<FormElement>**	Retrieves a list of all metadata group elements definition for a specific version of a node.

The grpName should be a valid Metadata group name.

The method is usually used to display empty form elements for updating new metadata values.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (FormElement fElement : ws.propertyGroup.getPropertyGroupByVersionFor
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getPropertyGroup

Description:

Method	Return values	Description
getPropertyGroup(String grpName)	**PropertyGroup**	Retrieve the metadata group

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.propertyGroup.getPropertyGroup("okg:consulting"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValue

Description:

Method	Return values	
getSuggestBoxKeyValue(String grpName, String propertyName, String key)	**String**	Retu

The propertyName parameter should be a Metadata Suggestbox field type.

More information at Metadata Suggestbox field

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                                 "http://www.openkm.com/dtd/property-groups-
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
     <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
       filterQuery="select ct_id, ct_name from country where ct_name like '
       valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.propertyGroup.getPropertyGroup("okg:consulting"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValuesFiltered

Description:

Method	Return values	
getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)	**Map<String, String>**	Retrieves a ma values

The propertyName parameter should be a Metadata Suggestbox field type.

More information at Metadata Suggestbox field

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                                 "http://www.openkm.com/dtd/property-groups
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
     <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
       filterQuery="select ct_id, ct_name from country where ct_name like '
       valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> values = ws.propertyGroup.getSuggestBoxKeyValuesFilte
            for (String key : values.keySet()) {
                String value = values.get(key);
                System.out.println(key + ":" + value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

validateField

Method	Return values
validateField(String value, String className, List<String> uuids)	**String**

- The "className" value must be the canonical class name of a class which implements FieldValidator interf

Example of the Form validator implementation

More information at Creating your own Form Validator plugin.

In this example it will not be allowed two equal comments in the metadata field named okp:tecnology.comment

```
package com.openkm.plugin.form.validator;

import java.util.ArrayList;
import java.util.List;

import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.form.FieldValidator;
import org.springframework.beans.factory.annotation.Autowired;

import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.db.service.LegacySrv;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements

    @Autowired
    private LegacySrv legacySrv;

    @Autowired
    private OKMRelation okmRelation;

    @Override
    public String getName() {
        return "Duplicated document number";
    }

    @Override
    public String validate(String value, List<String> uuids) {
        String validate = "";

        String token = DbSessionManager.getInstance().getSystemToken();
        String sql = "SELECT RGT_UUID FROM OKM_PGRP_CUR_TECHNOLOGY WHERE RG'
        try {
            List<List<String>> result = legacySrv.executeSQL(sql);
            if (result.size() > 0) {
                if (uuids.isEmpty()) {
                    validate = "Duplicated document number";
                } else {
                    List<String> allowedUuids = new ArrayList<>();
                    for (String uuid : uuids) {
                        allowedUuids.add(uuid);
                        List<Relation> relations = okmRelation.getRelations
                        for (Relation relation : relations) {
                            allowedUuids.add(relation.getNode());
                        }
                    }

                    for (List<String> resultList : result) {
                        if (!allowedUuids.contains(resultList.get(0))) {
                            validate = "Duplicated document number";
                        }
                    }
                }
            }
        } catch (Exception e) {
```

```
                validate = e.getMessage();
            }

            return validate;
        }
    }
```

Example

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<String> uuids = new ArrayList<>();
            uuids.add("26ece92f-9708-4d3f-8033-18dc98b9e7f5");
            uuids.add("7984b622-7c5f-425a-9451-7611c0caf227");

            String value = "test";
            String className = "com.openkm.plugin.form.validator.DuplicateDocumentNum
            String message = ws.propertyGroup.validateField(value, className, uuids);
            System.out.println("Validate: " + message);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Record samples

Basics

Example of uuid:

- Using UUID -> "**a66664a3-0e1d-4b03-9049-a2f4732a0802**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Record methods from " **record**" class as is shown below:

```
ws.record.createRecord("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new tit
```

Methods

createRecord

Description:

Method	Return values	Description
createRecord(String uuid, String name, String title, long nodeClass)	**Record**	Creates a new record and return as a result an object Record.
The parameters **uuid** should be any valid folder or record **UUID**.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Record record = ws.record.createRecord("ada67d44-b081-4b23-bdc1-74181caf
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordProperties

Description:

Method	Return values	Description
getRecordProperties(String uuid)	**Record**	Returns the record properties.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.record.getRecordProperties("fbe2933e-b141-4557-ab7a
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRecord

Description:

Method	Return values	Description
deleteRecord(String uuid)	**void**	Deletes a record.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.deleteRecord("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeRecord

Description:

Method	Return values	Description
purgeRecord(String uuid)	**void**	The record is definetivelly removed from repository.

Usually you will purge records to /okm:trash/userId - the personal trash user locations - but is possible to directly purge any record from the whole repository.

When a record is purged it will only be able to be restored from a previously repository backup. The purge action remove the record definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.purgeRecord("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

renameRecord

Description:

Method	Return values	Description
renameRecord(String uuid, String newName)	**Record**	Renames a record.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.renameRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "new_name"
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

moveRecord

Description:

Method	Return values	Description

moveRecord(String uuid, String dstId)	**void**	Moves a record to a folder or record.
The values of the dstId parameter should be a folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.moveRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyRecord

Description:

Method	Return values	Description
copyRecord(String uuid, String dstId, String newName)	**void**	Copies a record to a folder or record.

The values of the dstId parameter should be a folder or record UUID.

When the parameter newName value is null, the record will preserve the same name.

Only the security grants are copied to the destination, the metadata, keywords, etc. of the record are not copied.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.copyRecord("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValidRecord

Description:

Method	Return values	Description
isValidRecord(String uuid)	**Boolean**	Returns a boolean that indicate if the node is a record or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("Is a record:" + ws.record.isValidRecord("5489fc37-3eb
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordChildren

Description:

Method	Return values	Description
getRecordChildren(String uuid)	**List<Record>**	Returns a list of all records which their parent is fldId

The parameter uuid can be a folder or a record node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (Record rec : ws.record.getRecordChildren("8599eab7-ae61-4628-8010-11
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

lockRecord

Description:

Method	Return values	Description
lockRecord(String uuid)	**LockInfo**	Locks a record and return an object with the Lock information

Only the user who locked the record is allowed to unlock.

A locked record can not be modified by other users.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
```

```
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.lockRecord("5489fc37-3eb7-43de-998c-319725ae0ca0");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

unlockRecord

Description:

Method	Return values	Description
unlockRecord(String uuid)	**void**	Unlock a locked record.

Only the user who locked the document is allowed to unlock.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.unlockRecord("5489fc37-3eb7-43de-998c-319725ae0ca0");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceUnlockRecord

Description:

Method	Return values	Description

forceUnlockRecord(String uuid)	**void**	Unlocks a locked record.

This method allows to unlock any locked record.

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:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.forceUnlockRecord("5489fc37-3eb7-43de-998c-319725ae0ca0");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRecordTitle

Description:

Method	Return values	Description
setRecordTitle(String uuid, String title)	**void**	Sets a record title.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.setRecordTitle("5489fc37-3eb7-43de-998c-319725ae0ca0", "some ti
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRecordPath

Description:

Method	Return values	Description
getRecordPath(String uuid)	**String**	Converts a record UUID to a record path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.record.getRecordPath("5489fc37-3eb7-43de-998c-31972
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRecordNodeClass

Description:

Method	Return values	Description
setRecordNodeClass(String uuid, long ncId)	**void**	Set the NodeClass.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long ncId = 2;
            ws.record.setRecordNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRecordDispositionStage

Description:

Method	Return values	Description
setRecordDispositionStage(String uuid, long stage)	**void**	Set the disposition stage

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stage = 1;
            ws.record.setRecordDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRecordDescription

Description:

Method	Return values	Description
setRecordDescription(String uuid, String description)	**void**	Set a description.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.setRecordDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "s
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedRecordCopy

Description:

Method	Return values	Description
extendedRecordCopy(String uuid, String dstId, String newName, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)	**Record**	Copies a record with the associated data into some folder or record.

The values of the dstId parameter should be a folder or record UUID.

When parameter newName value is null, the record will preservate the same name.

By default only the binary data and the security grants, the metadata, keywords, etc. of the folder are not copyed.

Additional:

- When the category parameter is true the original values of the categories will be copied.

- When the keywords parameter is true the original values of the keywords will be copied.
- When the propertyGroups parameter is true the original values of the metadata groups will be copied.
- When the notes parameter is true the original values of the notes will be copied.
- When the security parameter is true the original value of the security will be copied.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Record record = ws.record.extendedRecordCopy("ada67d44-b081-4b23-bdc1-741
                    "new name record", true, true, true, true, true);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizardRecord

Description:

Method	Return values	Description
createWizardRecord(String uuid, String name, String title, long nodeClass)	**WizardNode**	Create a new record with wizard.

The parameters **uuid** should be any valid folder or record **UUID**.

The WizardNode contains a list of pending actions what should be done to complete the process of record creation. These might be:

- Add keyword

- Add Categories
- Add Metadata

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            WizardNode wn = ws.record.createWizardRecord("1f323e88-64ee-4f57-91e2-927
            System.out.print(wn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceLockRecord

Description:

Method	Return values	Description
forceLockRecord(String uuid)	**LockInfo**	Locks a record and returns an object with the Lock information.

This method allows to lock any record.

This action can only be done by a super user (user with ROLE_ADMIN).

In case exist a previous lock it will be replaced by a new one. When parent is locked you must apply the lock from parent.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.LockInfo;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            LockInfo lockInfo = ws.record.forceLockRecord("1ec49da9-1746-4875-ae32-92
            System.out.println("Author:" + lockInfo.getOwner());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createRecordFromTemplate

Description:

Method	Return values	Description
createRecordFromTemplate(String uuid, String dstPath, boolean categories, boolean keywords, boolean notes, Map<String, String> properties)	**Record**	Creates a new record from the template and returns an object Record.

The **uuid** parameter is the UUID value of the template file.

The **dstPath** value is the record destination path.

When the template uses metadata groups to fill in fields, then these values are mandatory and must be set into properties parameter.

Additional:

- When category parameter is true the original values of the categories will be copied.
- When keywords parameter is true the original values of the keywords will be copied.
- When notes parameter is true the original values of the notes will be copied.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Record;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            Map<String, String> properties = new HashMap<>();
            // okg:tpl
            properties.put("okp:tpl.name", "Some name");
            Calendar cal = Calendar.getInstance();
            // Value must be converted to String ISO 8601 compliant
            properties.put("okp:tpl.bird_date", ISO8601.formatBasic(cal));
            properties.put("okp:tpl.language", "[ \"java\" ]");

            // Property okg:technology
            properties.put("okp:technology.comment", "sdk name");
            Record record = ws.record.createRecordFromTemplate("9a114b17-7e51-41e7-9d
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createMissingRecords

Description:

Method	Return values	Description
createMissingRecords(String recPath)	**String**	Create missing records.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.record.createMissingRecords("/okm:root/missingrec1/missingrec2/missing
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Relation samples

Basics

Example of uuid:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";

Suggested code sample

First, you must create the web service object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should log in using the method "**login**." You can access the "**login**" method from the web service object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once logged in with the web services, the session is kept in the web service Object. Then you can use the other API method.

At this point, you can use all the Relation methods from the " **relation**" class as shown below:

```
ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL);
```

Methods

getRelationTypes

Description:

Method	Return values	Description
getRelationTypes(String type)	**List<RelationType>**	Retrieves a list of all relations defined of a type.
Available types values: • RelationType.BIDIRECTIONAL • RelationType.PARENT_CHILD • RelationType.MANY_TO_MANY		

More information on Relation types.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationType type : ws.relation.getRelationTypes(RelationType.BIDIRE
                System.out.println(type);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addRelation

Description:

Method	Return values	Description
addRelation(String nodeAId, String nodeBId, long relTypeId)	**void**	Sets a relation between two nodes.
The parameters **nodeAId** and **nodeBId** should be any valid document, folder, mail, or record **UUID**.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationType type : ws.getRelationTypes(RelationType.BIDIRECTIONAL))
                // looking for a relation named invoice
                if (type.getTitle().equals("invoice")) {
                    // Relation invoice with budget
                    ws.relation.addRelation("46762f90-82c6-4886-8d21-ad3017dd78a7", "
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRelation

Description:

Method	Return values	Description
deleteRelation(long relationId)	**void**	Deletes a relationship.

Only when a node will not use the relationship it can be deleted. Otherwise, you'll get an error.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Relation;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (Relation relation : ws.getRelations("46762f90-82c6-4886-8d21-ad3017d
                // looking for a relation named invoice
                if (relation.getRelationTitle().equals("invoice")) {
                    ws.relation.deleteRelation(relation.getId());
                }
            }
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }
}
```

getRelations

Description:

Method	Return values	Description
getRelations(String uuid)	**List<Relation>**	Retrieves a list of all relations of a node.
The parameter **uuid** should be any valid document, folder, mail, or record **UUID**.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Relation;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (Relation relation : ws.relation.getRelations("46762f90-82c6-4886-8d2
                System.out.println(relation);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRelationGroups

Description:

Method	Return values	Description
getRelationGroups(String uuid)	**List<RelationGroup>**	Retrieves a list of all related groups of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationGroup rg : ws.relation.getRelationGroups("46762f90-82c6-4886
                System.out.println(rg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addRelationGroup

Description:

Method	Return values	Description
addRelationGroup(String uuid, String groupName, long type)	**void**	Adds a relation group at a node.

A relation group only has the sense to apply a relation type of RelationType.MANY_TO_MANY.

More information on Relation types.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
```

```
            for (RelationType type : ws.relation.getRelationTypes(RelationType.MANY_T
                if (type.getTitle().equals("staple")) {
                    ws.addRelationGroup("46762f90-82c6-4886-8d21-ad3017dd78a7", "stap
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addNodeToGroup

Description:

Method	Return values	Description
addNodeToGroup(String uuid, long groupId)	**void**	Adds a node to an existing relation group.

On a relation group, it only has the sense to apply the type RelationType.MANY_TO_MANY.

More information on Relation types.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationGroup rg : ws.relation.getRelationGroups("46762f90-82c6-4886
                if (rg.getName().equals("staple group")) {
                    ws.relation.addNodeToGroup("8f101a85-88e7-4abe-8175-c5fea3e17d8b"
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRelationGroup

Description:

Method	Return values	Description
deleteRelationGroup(long groupId)	**void**	Remove the relation group from a node

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationGroup rg : ws.relation.getRelationGroups("8f101a85-88e7-4abe
                if (rg.getName().equals("staple group")) {
                    ws.relation.deleteRelationGroup(rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findRelationGroup

Description:

Method	Return values	Description
findRelationGroup(long groupId)	**RelationGroup**	Finds a relation group by id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
```

```
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long groupId = 1;
            System.out.println(ws.relation.findRelationGroup(groupId));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setRelationGroupName

Description:

Method	Return values	Description
setRelationGroupName(long groupId, String groupName)	**void**	Changes the relation group name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long groupId = 1;
            ws.relation.setRelationGroupName(groupId, "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllRelationGroups

Description:

Method	Return values	Description
getAllRelationGroups(int relationTypeId, String filter, int offset, int limit)	**RelationGroupResultSet**	Retrieves a list of all related groups.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" limits the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example, if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20; you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.bean.RelationGroupResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            int relationTypeId = 2;
            String filter = "";
            RelationGroupResultSet resultSet = ws.relation.getAllRelationGroups(relat
            System.out.println("Total: " + resultSet.getTotal());
            for (RelationGroup relationGroup : resultSet.getResults()) {
                System.out.println(relationGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteRelationGroupItem

Description:

Method	Return values	Description
deleteRelationGroupItem(String uuid, long groupId)	**void**	Remove a node from a related group.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.RelationGroup;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (RelationGroup rg : ws.relation.getRelationGroups("930baee0-8712-41b0
                if (rg.getName().equals("staple")) {
                    ws.relation.deleteRelationGroupItem("930baee0-8712-41b0-85c0-81d2
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Repository samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Repository methods from "**repository**" class as is shown below:

```
ws.repository.getAppVersion();
```

Methods

getRootFolder

Description:

Method	Return values	Description
getRootFolder()	**Folder**	Returns the object Folder of node "/okm:root"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

```
        try {
            ws.login(user, password);
            System.out.println(ws.repository.getRootFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTrashFolder

Description:

Method	Return values	Description
getTrashFolder()	**Folder**	Returns the object Folder of node "/okm:trash/{userId}"

The returned folder will be the user trash folder.

For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:trash/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getTrashFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTrashFolderBase

Description:

Method	Return values	Description

getTrashFolderBase()	**Folder**	Returns the object Folder of node "/okm:trash"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getTrashFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTemplatesFolder

Description:

Method	Return values	Description
getTemplatesFolder()	**Folder**	Returns the object Folder of node "/okm:templates"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getTemplatesFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalFolder

Description:

Method	Return values	Description
getPersonalFolder()	**Folder**	Returns the object Folder of node "/okm:personal/{userId}"

The returned folder will be the user personal folder.

For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:personal/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getPersonalFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalFolderBase

Description:

Method	Return values	Description
getPersonalFolderBase()	**Folder**	Returns the object Folder of node "/okm:personal"

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getPersonalFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMailFolder

Description:

Method	Return values	Description
getMailFolder()	**Folder**	Returns the object Folder of node "/okm:mail/{userId}"

The returned folder will be the user mail folder.

For example if the method is executed by the user "okmAdmin" then the folder returned will be "/okm:mail/okmAdmin".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getMailFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

getMailFolderBase

Description:

Method	Return values	Description
getMailFolderBase()	**Folder**	Returns the object Folder of node "/okm:mail"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getMailFolderBase());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getThesaurusFolder

Description:

Method	Return values	Description
getThesaurusFolder()	**Folder**	Returns the object Folder of node "/okm:thesaurus"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getThesaurusFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategoriesFolder

Description:

Method	Return values	Description
getCategoriesFolder()	**Folder**	Returns the object Folder of node "/okm:categories"

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getCategoriesFolder());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purgeTrash

Description:

Method	Return values	Description
purgeTrash()	**void**	Definitively removes from repository all nodes to "/okm:trash/{userId}"

For example if the method is executed by the user "okmAdmin" then the purged trash will be

"/okm:trash/okmAdmin".

When a node is purged it will only be able to be restored from a previously repository backup. The purge action removes the node definitely from the repository.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.repository.purgeTrash();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getUpdateMessage

Description:

Method	Return values	Description
getUpdateMessage()	**String**	Retrieves a message when there is a new OpenKM release.

There's an official OpenKM update message service available which is based on your local OpenKM version.

The most common message is that a new OpenKM version has been released.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getUpdateMessage());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRepositoryUuid

Description:

Method	Return values	Description
getRepositoryUuid()	**String**	Retrieves the installation unique identifier.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getRepositoryUuid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasNode

Description:

Method	Return values	Description
hasNode(String nodeId)	**Boolean**	Returns a node that indicate if a node exists or not.

The value of the parameter nodeId can be a valid **UUID** or **path**.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("Exists node:" + ws.repository.hasNode("373bcdd0-c082-
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodePath

Description:

Method	Return values	Description
getNodePath(String uuid)	String	Converts a node UUID to path.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getNodePath("373bcdd0-c082-4e7b-addd-e10
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(String nodePath)	**String**	Converts a node path to UUID.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getNodeUuid("/okm:root/test"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAppVersion

Description:

Method	Return values	Description
getAppVersion()	**AppVersion**	Returns information about OpenKM version.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getAppVersion());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copyAttributes

Description:

Method	Return values	Description
copyAttributes(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes)	**void**	Copy attributes from a node to other.

The values of the dstId parameter should be a node UUID.

- When the category parameter is true the original values of the categories will be copied.
- When the keywords parameter is true the original values of the keywords will be copied.
- When the property Groups parameter is true the original values of the metadata groups will be copied.
- When the notes parameter is true the original values of the notes will be copied.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.repository.copyAttributes("46762f90-82c6-4886-8d21-ad3017dd78a7", "ac9
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

executeScript

Description:

Method	Return values	Description
executeScript(InputStream is)	**ScriptExecutionResult**	Executes an script.

The local script - test.bsh - used in the sample below:

```
import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;

try {
    OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
    for (Folder fld : okmFolder.getChildren(null, "/okm:root")) {
        print(fld.getPath() + "\n");
    }
} catch (Exception e) {
    e.printStackTrace();
}

// Some value can also be returned as string
return "test result";
```

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ScriptExecutionResult;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
```

```
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/test.bsh");
            ScriptExecutionResult result = ws.repository.executeScript(is);
            System.out.println(result.getResult());
            System.out.println(result.getStdout());

            if (!result.getStderr().isEmpty()) {
                System.out.println("Error happened");
                System.out.println(result.getStderr());
            }

            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeScript

Description:

Method	Return values	Description
executeScript(String script)	**ScriptExecutionResult**	Executes an script.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ScriptExecutionResult;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String script = "import com.openkm.util.ContextWrapper;"
                    + " import org.springframework.web.context.WebApplicationContext;
                    + " import com.openkm.api.OKMFolder;"
                    + " import com.openkm.bean.Folder;"
                    + " WebApplicationContext cc = (WebApplicationContext) ContextWra
```

```
                    + " OKMFolder okmFolder = cc.getBean(OKMFolder.class);"
                    + " for (Folder fld : okmFolder.getChildren(null,\"/okm:root\"))"
                    + " print(fld.getPath());"
                    + "}";
            ScriptExecutionResult result = ws.repository.executeScript(script);
            System.out.println(result.getResult());
            System.out.println(result.getStdout());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeSqlQuery

Description:

Method	Return values	Description
executeSqlQuery(InputStream is)	**SqlQueryResults**	Executes SQL sentences.

The test.sql script used in the sample below:

```
SELECT NBS_UUID, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;
```

The SQL script can only contains a single SQL sentence.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlQueryResultColumns;
import com.openkm.sdk4j.bean.SqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/test.sql");
```

```
            SqlQueryResults result = ws.repository.executeSqlQuery(is);
            for (SqlQueryResultColumns columns : result.getResults()) {
                System.out.println("UUID:" + columns.getColumns().get(0));
                System.out.println("Context:" + columns.getColumns().get(1));
                System.out.println("Name:" + columns.getColumns().get(2));
            }
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Also the InputStream can be set as:

```
String sql = "SELECT NBS_UUID, NBS_CONTEXT, NBS_NAME FROM OKM_NODE_BASE LIMIT 10;";
InputStream is = new ByteArrayInputStream(sql.getBytes("UTF-8"));
```

executeSqlQuery

Description:

Method	Return values	Description
executeSqlQuery(String sql)	**SqlQueryResults**	Executes SQL sentences.

The SQL script can only contains a single SQL sentence.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlQueryResultColumns;
import com.openkm.sdk4j.bean.SqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            SqlQueryResults result = ws.repository.executeSqlQuery("SESELECT NBS_UUID
            for (SqlQueryResultColumns columns : result.getResults()) {
                System.out.println("UUID:" + columns.getColumns().get(0));
                System.out.println("Context:" + columns.getColumns().get(1));
                System.out.println("Name:" + columns.getColumns().get(2));
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeHqlQuery

Description:

Method	Return values	Description
executeHqlQuery(InputStream is)	**HqlQueryResults**	Executes HQL sentences.

The test.sql script used in the sample below:

```
SELECT uuid, author from NodeBase where name = 'okm:root';
```

The HQL script can only contains a single HQL sentence.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.HqlQueryResultColumns;
import com.openkm.sdk4j.bean.HqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/test.sql");
            HqlQueryResults result = ws.repository.executeHqlQuery(is);

            for (HqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid: " + row.getColumns().get(0) + ", name: " +
            }
```

```
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Also the InputStream can be set as:

```
String sql = "SELECT uuid, name from NodeBase where name = 'okm:root';";
InputStream is = new ByteArrayInputStream(sql.getBytes("UTF-8"));
```

executeHqlQuery

Description:

Method	Return values	Description
executeHqlQuery(String hql)	**HqlQueryResults**	Executes HQL sentences.

The HQL script can only contains a single HQL sentence.

This action can only be done by a super user (user with ROLE_ADMIN).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.HqlQueryResultColumns;
import com.openkm.sdk4j.bean.HqlQueryResults;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            HqlQueryResults result = ws.repository.executeHqlQuery("SELECT uuid, auth
            for (HqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid: " + row.getColumns().get(0) + ", name: " +
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTranslations

Description:

Method	Return values
getTranslations(String lang, String module)	**Map<String, String>**

The OpenKM translations tables can be used to retrieve actually OpenKM translations or create your own translatio

By default modules values are :

- frontend (used by default OpenKM UI).
- extension (used by OpenKM extension UI).
- mobile (used by OpenKM mobile UI).

Example to add a new Translation module :

SQL values to be executed from Database query view:

```
DELETE FROM OKM_TRANSLATION WHERE TR_LANGUAGE='en-GB' and TR_MODULE='doc';
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
```

The code then should be:

```
Map<String, String> translations = ws.getTranslations("en-GB", "doc");
```

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> translations = ws.repository.getTranslations("en-GB",
            for (String key : translations.keySet()) {
                System.out.println("key:'" + key + "', with translation:'" + translat
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getConfiguration

Description:

Method	Return values	Description
getConfiguration(String key)	**Configuration**	Retrieve the value of a configuration parameter.

If your OpenKM version have the configuration parameter named "**webservices.visible.properties**", will be restricted for non Administrator users what parameters are accessible. That means any non Administrator use who will try accessing across the webservices to configuration parameters not set into the list of values of "**webservices.visible.properties**" will get an access denied exception.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Configuration;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Configuration configuration = ws.repository.getConfiguration("system.ocr"
            System.out.println(configuration);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChangeLog

Description:

Method	Return values	Description

getChangeLog(String nodePath, Calendar modificationsFrom)	**List<ChangeLogged>**	Return the list of changes in some path and subfolders.

- The method is used by desktop synchronization application for retrieving the changes.

Example:

```
package com.openkm;

import java.util.Calendar;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.ChangeLogged;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Calendar now = Calendar.getInstance();
            for (ChangeLogged cl : ws.repository.getChangeLog("/okm:root/synchronized
                System.out.println(cl);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getServerTime()

Description:

Method	Return values	Description
getServerTime()	**String**	Return the current server time.

The server time returned format is ISO8601

- The method is used by desktop synchronization application for retrieving the changes

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println(ws.repository.getServerTime());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAvailableLocales

Description:

Method	Return values	Description
getAvailableLocales(String locale)	**Map<String, String>**	Return the available languages

Example:

```
package com.openkm;

import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> locales = ws.repository.getAvailableLocales("en-GB");
            for (String key : locales.keySet()) {
                System.out.println("Language:" + key + "," + locales.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getLicenseInfo()

Description:

Method	Return values	Description
getLicenseInfo()	**LicenseInfo**	Return license information.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.LicenseInfo;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            LicenseInfo licenseInfo = ws.repository.getLicenseInfo();
            System.out.println(licenseInfo);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getClusterUuid()

Description:

Method	Return values	Description
getClusterUuid()	String	Return cluster UUID.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String clusterUuid = ws.repository.getClusterUuid();
            System.out.println(clusterUuid);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getIsFilePlan()

Description:

Method	Return values	Description
getIsFilePlan()	boolean	True when the filePlan mode is active

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("FilePlan: " + ws.repository.getIsFilePlan());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Report samples

Basics

The table below shows how should be passed variables based on field type:

Field type	Type	Description
Date	**String**	Use the pattern yyyy-MM-dd (year - month - day) `2018-10-04`
Select multiple	**String**	Use "," to split each value `"value1,value2,value3"`

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Report methods from "**report**" class as is shown below:

```
ws.report.getReports(true)
```

Methods

getReports

Description:

Method	Return values	Description
getReports(boolean active)	**List<Report>**	Returns a list of reports.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (Report rep : ws.report.getReports(true)) {
                System.out.println(rep);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getReport

Description:

Method	Return values	Description
getReport(long rpId)	**Report**	Returns a reports.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Report rep = ws.report.getReport(1);
            System.out.println(rep);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

generateDownloadReportToken

Description:

Method	Return values	Description
generateDownloadReportToken(long rpId)	**String**	Return the token for downloading the report.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String token = ws.report.generateDownloadReportToken(1);
            System.out.println(token);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeReport

Description:

Method	Return values	Description
executeReport(long rpId, Map<String, String> params, String format, String uuid)	**InputStream**	Return a document result of executing a report.

Available formats:

- Report.FORMAT_CSV
- Report.FORMAT_DOCX

- Report.FORMAT_HTML
- Report.FORMAT_ODT
- Report.FORMAT_PDF
- Report.FORMAT_RTF
- Report.FORMAT_TEXT

The parameter **uuid** is the UUID of a node (this parameter is optional, usually has sense with reports executed from user interface where the results of the reports depend on the node selected).

Example:

```
package com.openkm;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> params = new HashMap<>();
            params.put("from_date", "2021-01-01");
            params.put("to_date", "2021-05-01");

            long rpId = 1;
            String format = Report.FORMAT_PDF;
            String dstId = "c5c87bd3-0c03-4bba-ab2f-7184c23a26d8";
            String uuid = "";
            InputStream is = ws.report.executeReport(rpId, params, format, dstId, uui
            OutputStream fos = new FileOutputStream("/home/openkm/report/document Cre
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveReport

Description:

Method	Return values	Description
saveReport(long rpId, Map<String, String> params, String format, String dstId, docName, String uuid)	Document	Execute the report and save the resulting document.

Available formats:

- Report.FORMAT_CSV
- Report.FORMAT_DOCX
- Report.FORMAT_HTML
- Report.FORMAT_ODT
- Report.FORMAT_PDF
- Report.FORMAT_RTF
- Report.FORMAT_TEXT

The values of the **dstld** parameter should be a folder or record UUID.

The parameter **docName** is the file name of the report.

The parameter **uuid** is the UUID of a node (this parameter is optional, usually has sense with reports executed from user interface where the results of the reports depend on the node selected).

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.bean.Report;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.HashMap;
import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, String> params = new HashMap<>();
            params.put("from_date", "2021-01-01");
            params.put("to_date", "2021-05-01");

            long rpId = 1;
```

```
            String format = Report.FORMAT_PDF;
            String dstId = "c5c87bd3-0c03-4bba-ab2f-7184c23a26d8";
            String docName = "Document create.pdf";
            String uuid = "";
            Document documenReport = ws.report.saveReport(rpId, params, format, dstId
            System.out.println(documenReport);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Stamp samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Stamp methods from "**stamp**" class as is shown below:

```
ws.stamp.getAllStamps();
```

Methods

getAllStamps

Description:

Method	Return values	Description
getAllStamps()	**List<StampItem>**	Returns a list of the stamp items

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampItem;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<StampItem> result = ws.stamp.getAllStamps();
            for (StampItem stampItem : result) {
                System.out.println(stampItem.getName());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampTextByPk

Description:

Method	Return values	Description
getStampTextByPk(long id, String uuid)	**StampText**	Return the stamp of type Text by Id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampText;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            int stampTextId = 1;
            StampText stampText = ws.stamp.getStampTextByPk(stampTextId, "928de61e-6c
            System.out.println(stampText);
            System.out.println(stampText.getName());
            System.out.println(stampText.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampImageByPk

Description:

Method	Return values	Description

getStampImageByPk(long id, String uuid)	**StampImage**	Return the stamp of type image by Id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampImage;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            int stampImageId = 3;
            StampImage stampImage = ws.stamp.getStampImageByPk(stampImageId, "928de61
            System.out.println(stampImage);
            System.out.println(stampImage.getName());
            System.out.println(stampImage.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStampBarcodeByPk

Description:

Method	Return values	Description
getStampBarcodeByPk(long id, String uuid)	**StampBarcode**	Return the stamp of type barcode by Id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampBarcode;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

```
        try {
            ws.login(user, password);
            int stampBarcodeId = 3;
            StampBarcode stampBarcode = ws.stamp.getStampBarcodeByPk(stampBarcodeId,
            System.out.println(stampBarcode);
            System.out.println(stampBarcode.getName());
            System.out.println(stampBarcode.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

calculateStampCoordinates

Description:

Method	Return values	Description
calculateStampCoordinates(long imgToStampWidth, long imgToStampHeight, long floatingDivWidth, long floatingDivHeight, String exprX, String exprY, String stampType, String stampAlign)	**StampCoordinates**	Return the calculated stamp coordinates.

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	`PAGE_CENTER`
Expr. Y	`PAGE_MIDDLE`

The parameter **stampAlign.**

Available values:

- text
- image

The parameter **stampAlign** is the Text Align

Available values:

- left
- center
- right

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampCoordinates;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingDivWidth = 250;
            long floatingDivHeight = 300;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            String stampType = "text";
            String stampAlign = "center";
            StampCoordinates stampCoordinates = ws.stamp.calculateStampCoordinates(im
                    floatingDivWidth, floatingDivHeight, exprX, exprY, stampType, sta
            System.out.println("X = " + stampCoordinates.getX() + ", Y = " + stampCoo

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

calculateStampExpressions

Description:

Method	Return values	Description
calculateStampExpressions(long imgToStampWidth, long		Return the calculated

imgToStampHeight, long posX, long posY)	**StampExpressions**	stamp expressions for X and Y.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampExpressions;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long posX = 10;
            long posY = 50;
            StampExpressions stampExpressions = ws.stamp.calculateStampExpressions(im
            System.out.println("X = " + stampExpressions.getExprX() + ", Y = " + stam

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampText

Description:

Method	Return values	Description
stampText(String uuid, long id)	**void**	Stamp text in a document.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stampTextId = 1;
            ws.stamp.stampText("babe24df-c443-49a5-9453-39dfc9b27924", stampTextId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImage

Description:

Method	Return values	Description
stampImage(String uuid, long id)	**void**	Stamp image in a document.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stampImageId = 1;
            ws.stamp.stampImage("babe24df-c443-49a5-9453-39dfc9b27924", stampImageId)
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampBarcode

Description:

Method	Return values	Description
stampBarcode(String uuid, long id)	**void**	Stamp barcode in a document.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stampBarcodeId = 1;
            ws.stamp.stampBarcode("f02053e3-2264-4040-bb84-67983a0208f7", stampBarcod
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImageManually

Description:

Method	Return values	Description
stampImageManually(String uuid, long id, String exprX, String exprY, String range, String personalStampUuid)	**void**	Stamp in a document with an image that is in openkm

The **id** is the id of an image stamp which is created in the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

The **personalStampUuid** must be a valid document **UUID**. The document must be an image (jpg or png).

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long id = 1;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            ws.stamp.stampImageManually("055b5206-35d0-4351-ba92-c304bbba7ddf", id, e
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampTextCustom

Description:

Method	Return values	Description
stampTextCustom(String uuid, long id, String text, String exprX, String exprY, String range)	**void**	Stamp in a document with a custom text.

The **id** is the id of an image stamp which is created in the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	`PAGE_CENTER`
Expr. Y	`PAGE_MIDDLE`

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.io.FileInputStream;
import java.io.InputStream;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stampTextId = 1;
            String text = "OpenKM";
            String exprX = "PAGE_CENTER";
            String exprY = "PAGE_MIDDLE";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
            ws.stamp.stampTextCustom("05b14eee5-2e57-4d1d-925a-c9bc3f526e24", stampTe
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

stampImageCustom

Description:

Method	Return values	Description

stampImageCustom(String uuid, long id, String exprX, String exprY, String range, double ratio, InputStream is)	**void**	Stamp in a document with a custom image.

The **id** is the id of an image stamp which is created in the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	`PAGE_CENTER`
Expr. Y	`PAGE_MIDDLE`

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.io.FileInputStream;
import java.io.InputStream;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long id = 1;
            String exprX = "PAGE_CENTER - IMAGE_WIDTH / 2";
            String exprY = "PAGE_MIDDLE - IMAGE_HEIGHT / 2";
            InputStream is = new FileInputStream("/home/gnujavasergio/okm/logo.png");
```

```
            ws.stamp.stampImageCustom("055b5206-35d0-4351-ba92-c304bbba7ddf", id, exp
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

calculateFlyImageDimensions

Description:

Method	Return values	Description
calculateFlyImageDimensions(String uuid, long imgToStampWidth, long imgToStampHeight, long floatingImageWidth, long floatingImageHeight, int pageNumber)	**StampImageSize**	Return the calculated image to stamp height and width size.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampImageSize;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Parameters
            long imgToStampWidth = 100;
            long imgToStampHeight = 100;
            long floatingImageWidth = 300;
            long floatingImageHeight = 300;
            int pageNumber = 2;

            StampImageSize stampImageSize = ws.stamp.calculateFlyImageDimensions("bab
            System.out.println(stampImageSize);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalStamps

Description:

Method	Return values	Description
getPersonalStamps()	**List<Document>**	Returns a list of personal seals.

Return a list of documents of type image (jpg or png) from /okm:templates/ userId/stamps

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<Document> result = ws.stamp.getPersonalStamps();
            for (Document doc : result) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalStampImage

Description:

Method	Return values	Description
getPersonalStampImage(String uuid, long id)	**StampPersonalImage**	Return the personal seal processed.

When stamping a document with an image, the image also can have a text layer.

This method processes the image and returns an image with the text layer.

The **uuid** must be an **uuid** returned by the method getPersonalStamps.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.StampPersonalImage;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long stampId = 1;
            StampPersonalImage stampPersonalImage = ws.stamp.getPersonalStampImage("9
            System.out.println(stampPersonalImage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Search samples

Basics

Almosts all methods use QueryParams. Here there're some tips about how using it.

Variables	Type	Allow wildcards	Restr
domain	**long**	No.	Available values: • QueryParams.DOCUMENT • QueryParams.FOLDER • QueryParams.MAIL • QueryParams.RECORD By default the value is set to QueryParams.DOCUMENT. For searching documents and folders use value: `(QueryParams.DOCUMENT | QueryParams.FOLDER)`
author	**String**	No.	Value must be a valid userId.
name	**String**	Yes.	
title	**String**	Yes.	
keywords	**Set<String>**	Yes.	
categories	**Set<String>**	No.	Values should be a category UUID, not use path value.

content		Yes.	
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	**Boolean**	No.	Only has sense to set this variable to true when the variable folder is not
lastModifiedFrom	**Calendar**	No.	
lastModifiedTo	**Calendar**	No.	
mailSubject	**String**	Yes.	Only apply to mail nodes.

mailFrom	**String**	Yes.	Only apply to mail nodes.
mailTo		Yes.	Only apply to mail nodes.
notes		Yes.	

properties	**Map<String, String>**	Yes on almost.	On metadata field values like "date" can not be applied wilcards.
			
			The map of the properties is composed of pairs:
			
			('metadata_field_name','metada_field_value")
			
			For example:
			
			`Map<String, String> properties = new HashMap()`
			`properties.put("okp:consulting.name", "name va`
			
			Filtering by range of dates:
			
			`Calendar to = Calendar.getInstance(); // today`
			`to.set(0, Calendar.HOUR);`
			`to.set(0, Calendar.MINUTE);`
			`to.set(0, Calendar.SECOND);`
			`to.set(0, Calendar.MILLISECOND);`
			`Calendar from = (Calendar) to.clone();`
			`from.add(-3, Calendar.DATE); // three days bef`
			`Map<String,String> properties = new HashMap<>(`
			`properties.put("okp:consulting.date", ISO8601.`
			
			When filtering by range of dates you must set both values (side.
			
			To filtering by a metadata field of type multiple like this:
			
			`<select label="Multiple" name="okp:consulting.`
			`<option label="One" value="one"/>`
			`<option label="Two" value="two"/>`

```
  <option label="Three" value="three" />
</select>
```

You should do:

```
properties.put("okp:consulting.multiple", "one
```

Where "**one**" and "**two**" are valid values and character "**;**" is used as sep

The search operation is done only by AND logic.

Wildcard examples:

Variable	Example	Description
name	**test*.html**	Any document that starts with characters "test" and ends with characters ".html"
name	**test?.html**	Any document that starts with characters "test" followed by a single character and ends with characters ".html"
name	**?test***	Any of the documents where the first character doesn't matter, but is followed by the characters, "test".

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Search methods from "**search**" class as is shown below:

```
ws.search.find(qParams, null)
```

Methods

find

Description:

Method	Return values	Description
find(QueryParams queryParams, String propertiesPlugin)	**List<QueryResult>**	Returns a list of results filtered by the values of the queryParams parameter.

The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            for (QueryResult qr : ws.search.find(params, null)) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

find

Description:

Method	Return values	Description
find(QueryParams queryParams, String sortField, boolean sortReverse, String propertiesPlugin)	**List<QueryResult>**	Returns a list of results filtered by the values of the queryParams parameter.

The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            for (QueryResult qr : ws.search.find(params, SearchSortField.LAST_MODIFIE
                System.out.println(qr);
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated

Description:

Method	Return values	Description
findPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin)	**ResultSet**	Returns a list of paginated results filtered by the values of the queryParams parameter.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
```

```
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");
            ResultSet rs = ws.search.findPaginated(params, 20, 10, null);
            System.out.println("Total results: " + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPaginated

Description:

Method	Return values	Description
findPaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	**ResultSet**	Returns a list of paginated results filtered by the values of the queryParams parameter.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            ResultSet rs = ws.search.findPaginated(params, SearchSortField.LAST_MODIF
            System.out.println("Total results: " + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBasePaginated

Description:

Method	Return values	Description
findSimpleNodeBasePaginated(QueryParams queryParams, int offset, int limit)	**SimpleNodeBaseResultSet**	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods (these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enought. Specially when the limit is a higher value you will appreciate the difference in the perfomance.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
```

```
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");
            SimpleNodeBaseResultSet rs = ws.search.findSimpleNodeBasePaginated(params
            System.out.println("Total results: " + rs.getTotal());

            for (SimpleNodeBase sn : rs.getResults()) {
                System.out.println(sn);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBasePaginated

Description:

Method	Return values	Description
findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit)	**SimpleNodeBaseResultSet**	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods (these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enought. Specially when the limit is a higher value you will appreciate the difference in the perfomance.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBasePaginated(pa
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBasePaginated

Description:

Method	Return values	Description
		Returns a list of

findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit, String pluginName)	**SimpleNodeBaseResultSet**	SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods (these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enought. Specially when the limit is a higher value you will appreciate the difference in the perfomance.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");
            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBasePaginated(pa
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getKeywordMap

Description:

Method	Return values	Description
getKeywordMap(List<String> filter)	**Map<String, Integer>**	Returns a map of keywords with its count value filtered by other keywords.

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:

```
package com.openkm;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // All keywords without filtering
            System.out.println("Without filtering");
            Map<String, Integer> keywords = ws.getKeywordMap(new ArrayList<String>())

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key));
            }

            // Keywords filtered
            System.out.println("Filtering");
            keywords = ws.search.getKeywordMap(Arrays.asList("test"));

            for (String key : keywords.keySet()) {
                System.out.println(key + " is used :" + keywords.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCategorizedDocuments

Description:

Method	Return values	Description
getCategorizedDocuments(String categoryId)	**List<Document>**	Retrieves a list of all documents related with a category.
The values of the categoryId parameter should be a category folder UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (Document doc : ws.search.getCategorizedDocuments("58c9b25f-d83e-4006
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveSearch

Description:

Method	Return values	Description
saveSearch(QueryParams params)	**Long**	Saves a search parameters.
The variable queryName of the parameter params, should have to be initialized.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");
            for (QueryResult qr : ws.find(qParams, null)) {
                System.out.println(qr);
            }
            // Save the search to be used later
            qParams.setQueryName("sample search");
            ws.search.saveSearch(qParams);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateSearch

Description:

Method	Return values	Description
updateSearch(QueryParams params)	**void**	Updates a previously saved search parameters.

Only can be updated as a saved search created by the same user user who's executing the method.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (QueryParams qParams : ws.getAllSearchs()) {
                if (qParams.getQueryName().equals("sample search")) {
                    // Change some value.
                    qParams.setName("admin*.html");
                    ws.search.updateSearch(qParams);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSearch

Description:

Method	Return values	Description

getSearch(int qpId)	**QueryParams**	Gets saved search parameters.

Only can be retrieved as a saved search created by the same user who's executing the method.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            int qpId = 1; // Some valid search id
            QueryParams qParams = ws.search.getSearch(qpId);
            System.out.println(qParams);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllSearchs

Description:

Method	Return values	Description
getAllSearchs()	**List<QueryParams>**	Retrieves a list of all saved search parameters.

Only will be retrieved the list of the saved searches created by the same user who's executing the method.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (QueryParams qParams : ws.search.getAllSearchs()) {
                System.out.println(qParams);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteSearch

Description:

Method	Return values	Description
deleteSearch(int qpId)	**void**	Deletes a saved search parameters.

Only can be deleted as a saved search created by the same user user who's executing the method.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            int qpId = 1; // Some valid search id
            ws.search.deleteSearch(qpId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSearchConfig

Description:

Method	Return values	Description
getSearchConfig(String pluginName)	**NodeSearchConfig**	Return a NodeSearchConfig object.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.NodeSearchConfig;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            NodeSearchConfig nodeSearchConfig = ws.search.getSearchConfig("com.openkm
            System.out.println(nodeSearchConfig);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSearchPlugins

Description:

Method	Return values	Description
getSearchPlugins()	**SearchPluginList**	Return a SearchPluginList object.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SearchPlugin;
import com.openkm.sdk4j.bean.SearchPluginList;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            SearchPluginList pluginList = ws.search.getSearchPlugins();
            for (SearchPlugin searchPlugin : pluginList.getSearchPlugins()) {
                System.out.println(searchPlugin);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findAllDefaultByNodeClass

Description:

Method	Return values	Description
findAllDefaultByNodeClass(long ncId)	**List<QueryParams>**	Retrieve a list of saved searches of a NodeClass

Example:

```
package com.openkm;

import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long ncId = 1;
            List<QueryParams> results = ws.search.findAllDefaultByNodeClass(ncId);
            for (QueryParams params : results) {
                System.out.println(params);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQuery

Description:

Method	Return values	Description
findByQuery(String query, String propertiesPlugin)	**List<QueryResult>**	Returns a list of results filtered by the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<QueryResult> results = ws.search.findByQuery("keyword:test AND name:
            for (QueryResult qr : results) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQuery

Description:

Method	Return values	Description
findByQuery(String query, String sortField, boolean sortReverse, String propertiesPlugin)	**List<QueryResult>**	Returns a list of results filtered by the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.List;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<QueryResult> results = ws.search.findByQuery("keyword:test AND name:
            for (QueryResult qr : results) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	Return values	Description
findByQueryPaginated(String query, int offset, int limit, String propertiesPlugin)	**ResultSet**	Returns a list of paginated results filtered by the values of the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ResultSet rs = ws.search.findByQueryPaginated("text:grial AND name:t*.pdf
            System.out.println("Total results:" + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByQueryPaginated

Description:

Method	Return values	Description
findByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)	**ResultSet**	Returns a list of paginated results filtered by the values of the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

Available sortField values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ResultSet rs = ws.search.findByQueryPaginated("text:grial AND name:t*.pdf
            System.out.println("Total results:" + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findSimpleNodeBaseByQueryPaginated

Description:

Method	Return values	Description
findSimpleNodeBaseByQueryPaginated(String query, int offset, int limit)	**SimpleNodeBaseResultSet**	Returns a list of paginated results filtered by the values of the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SimpleNodeBase;
import com.openkm.sdk4j.bean.SimpleNodeBaseResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBaseByQueryPagin
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
            System.out.println("Total: " + result.getTotal());
        } catch (Exception e) {
```

```
                e.printStackTrace();
            }
        }
    }
```

findSimpleNodeBaseByQueryPaginated

Description:

Method	Return values	Description
findSimpleNodeBaseByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit)	**SimpleNodeBaseResultSet**	Returns a list of paginated results filtered by the values of the query parameter.

The **syntax** to use in the statement parameter is the pair **'field:value'**. For example:

- "name:grial" is filtering field name by word grial.

More information about lucene sintaxis at Lucene query syntax.

Available sortField values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            SimpleNodeBaseResultSet result = ws.search.findSimpleNodeBaseByQueryPagin
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadata

Description:

Method	Return values	Description
findWithMetadata(QueryParams queryParams, String propertiesPlugin, List<String> groups)	**List<QueryResult>**	Returns a list of results with metadata values filtered by the queryParams parameter.

- The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.
- The parameter "groups" must be valid metadata group names.

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");

            for (QueryResult qr : ws.search.findWithMetadata(qParams, null, groups))
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadata

Description:

Method	Return values	Description
findWithMetadata(QueryParams queryParams, String sortField, boolean sortReverse, String propertiesPlugin, List<String> groups)	**List<QueryResult>**	Returns a list of results with metadata values filtered by the queryParams parameter.

- The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.
- The parameter "groups" must be valid metadata group names.

Available sortField values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.ArrayList;
import java.util.List;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");
            for (QueryResult qr : ws.search.findWithMetadata(params, SearchSortField.
                System.out.println(qr);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMedataPaginated

Description:

Method	Return values	Description

findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List<String> groups)	**ResultSet**	Returns a list of paginated results with metadata values filtered by the queryParams parameter.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

The parameter "groups" must be valid metadata group names.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import java.util.ArrayList;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            qParams.setName("test*");

            List<String> groups = new ArrayList<>();
            groups.add("okg:consulting");

            ResultSet rs = ws.search.findWithMetadataPaginated(qParams, 0, 10, null,
            System.out.println("Total results: " + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMedataPaginated

Description:

Method	Return values	Description
findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List<String> groups)	**ResultSet**	Returns a list of paginated results with metadata values filtered by the queryParams parameter.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

The parameter "groups" must be valid metadata group names.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

For example if your query has 1000 results, but you only want to return the first 10, you should use these

values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

Retrieving entire Objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.bean.QueryResult;
import com.openkm.sdk4j.bean.ResultSet;
import com.openkm.sdk4j.bean.SearchSortField;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            ResultSet rs = ws.search.findWithMetadataPaginated(params, SearchSortFiel
            System.out.println("Total results: " + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getMimeTypes

Description:

Method	Return values	Description

getMimeTypes()	**List<MimeType>**	Retrieves a list of mimetypes.

Only will be retrieved the list of the mimeTypes what may be used in the search.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.MimeType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (MimeType mimeType : ws.search.getMimeTypes()) {
                System.out.println(mimeType);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

csvExport

Description:

Method	Return values	Description
csvExport(String token, String lang, QueryParams queryParams, boolean compact)	**InputStream**	Export as a csv a list of results filtered by the values of the queryParams parameter.

The parameter **lang** must be ISO 691-1 compliant.

More information at: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.QueryParams;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.apache.commons.io.IOUtils;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            QueryParams params = new QueryParams();
            params.setDomain(QueryParams.DOCUMENT + QueryParams.FOLDER);
            params.setName("test*");

            InputStream is = ws.search.csvExport("es-ES", params, false);
            OutputStream fos = new FileOutputStream("/home/openkm/okm/export.csv");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Shard samples

Basics

Example of uuid:

- Using UUID -> "**f123a950-0329-4d62-8328-0ff500fd42db**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Shard methods from "**shard**" class as is shown below:

```
ws.shard.getShards()
```

Methods

getShards

Description:

Method	Return values	Description
getShards()	**List<Shard>**	Returns a list of all shards

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Shard;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            for (Shard shard : ws.shard.getShards()) {
                System.out.println(shard);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setCurrentShard

Description:

Method	Return values	Description
setCurrentShard(long shardId)	**void**	Change the current shard.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            long shardId = 1;
            ws.shard.setCurrentShard(1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

changeShard

Description:

Method	Return values	Description
changeShard(String uuid, long shardId)	**void**	Change the shardId of the uuid

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String username = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(username, password);

            long shardId = 1;
            ws.shard.changeShard("e2391b01-ce10-42c7-94a9-b0d6b394048e", shardId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Task samples

Basics

Task fields description:

Field	Type	Description	Mandatory
Id	**Long**	Internal task id. It is automatically set by application.	Not applicable.
Owner	**String**	It contains the OpenKM userId. It is automatically set by application.	Not applicable.
Subject	**String**	The topic of the task.	Yes.
Description	**String**	The description of the task.	No.
Start	**Calendar**	When the task might start.	Yes.
End	**Calendar**	When the task might endding.	No.
Status	**TaskStatus**	The task status. You administering the list of status values.	Yes.
Project	**TaskProject**	The project related with the task. You administering the list of project values.	Yes.

Type	**TaskType**	The task type. You administering the list of task type values.	Yes.
Progress	**Integer**	The numeric progress status. The range of allowed values is from 0 to 100. Where 100% indicates a completed task.	Yes.
RepeatGroup	**Long**	When a task is repeated along time is a member of a group. The group is identified by an unique repeat group id.	No.
ReminderStartValue	**Integer**	How many days before the task starting, the system might send a mail notification to the user.	No.
ReminderEndValue	**Integer**	How many days before the task ending, the system might send a mail notification to the user.	No.
ReminderStartUnit	**String**	Reminder start units. Allowed values: • "m" for minutes. • "h" for hours. • "d" for days.	Mandatory when **ReminderStartValue** greater than 0.
ReminderEndUnit	**String**	Reminder end units. Allowed values: • "m" for minutes.	Mandatory when **ReminderEndValue** greater than 0.

		• "h" for hours. • "d" for days.	
Users	**Set<String>**	Collection of assigned users to the task.	No. But At least should be a user or role assigned.
Roles	**Set<String>**	Collection of assigned roles to the task.	No. But At least should be a user or role assigned.
Documents	**Set<String>**	List of UUID's of the related documents.	No.
Folders	**Set<String>**	List of UUID's of the related folders.	No.
Mails	**Set<String>**	List of UUID's of the related mails.	No.
Records	**Set<String>**	List of UUID's of the related records.	No.

On all methods, you'll see parameter named "**token**". When accessing application across SOAP the login process returns a token, what is used to identify the user on all the exposed methods. From default application execution context you must use the "**null**" value what indicates the application must use the "**user session**".

In special cases, you might be "**promoted as Administrator**" using the "**administrator token**".

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Task methods from " **task**" class as is shown below:

```
ws.task.getAssignedTasks(projectId, typeId, statusId, orderColumn, true, 0, 10, subje
```

Methods

getAssignedTasks

Description:

Method	Return values	Description
getAssignedTasks(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)	**List<Task>**	Retrieve a list of tasks assigned to a user, filtered and paginated.

Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optinal)
- The subject parameter the topic of the task.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these

values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            String orderColumn = "subject"; // A valid TaskManagerTask class paramete
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getAssignedTasks(projectId, typeId, statu
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActiveTasks

Description:

Method	Return values	Description
getActiveTasks(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)	**List<Task>**	Retrieve a list of active tasks assigned to a user, filtered and paginated.

Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optional)
- The subject parameter the topic of the task.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            String orderColumn = "subject"; // A valid TaskManagerTask class paramete
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getActiveTasks(projectId, typeId, statusI
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinishedTasks

Description:

Method	Return values	Description
getFinishedTasks(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)	**List<Task>**	Retrieve a list of finished tasks assigned to a user, filtered and paginated.

Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter is optional)
- The to date to filter (this parameter is optional)
- The subject parameter the topic of the task.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
import java.util.Calendar;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            String orderColumn = "subject"; // A valid TaskManagerTask class paramete
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getFinishedTasks(projectId, typeId, statu
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotifiedTasks

Description:

Method	Return values	Description
getNotifiedTasks(long projectId, long typeId, long statusId, String orderColumn, boolean orderAsc, int offset, int limit, Calendar from, Calendar to, String subject)	**List<Task>**	Retrieve a list of notified tasks assigned to a user, filtered and paginated.

Filter parameters description:

- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.
- The statusId parameters must be a valid status id.
- The orderColumn parameter must be a valid parameters of the TaskManagerTask class. Common

used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.

- The orderAsc parameter orders ascending or descending.
- The from date to filter (this parameter optional)
- The to date to filter (this parameter optional)
- The subject parameter the topic of the task.

The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query.

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.

For example if your query has 1000 results, but you only want to return the first 10, you should use these values:

- limit=10
- offset=0

Now suppose you want to show the results from 11-20, you should use these values:

- limit=10
- offset=10

The parameters "from" and "to" are optional and allow you to retrieve just a portion of the results of a query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.*;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Calendar;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
```

```
            String orderColumn = "subject"; // A valid TaskManagerTask class paramete
            Calendar from = Calendar.getInstance();
            from.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 1);

            Calendar to = Calendar.getInstance();
            to.set(from.get(Calendar.YEAR), Calendar.NOVEMBER, 30);

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            TaskList listAssigned = ws.task.getNotifiedTasks(projectId, typeId, statu
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskStatus

Description:

Method	Return values	Description
getTaskStatus()	**List<TaskStatus>**	Retrieve a list of all the task status.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskStatus ts : ws.task.getTaskStatus()) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskProjects

Description:

Method	Return values	Description
getTaskProjects(boolean filterActive)	**List<TaskProject>**	Retrieve a list of all the task projects.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskProject tp : ws.task.getTaskProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskTypes

Description:

Method	Return values	Description
getTaskTypes(boolean filterActive)	**List<TaskType>**	Retrieve a list of all the task types.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskType tt : ws.task.getTaskTypes(true)) {
                System.out.println(tt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAssignedTasksCount

Description:

Method	Return values	Description
getAssignedTasksCount(long statusId, long projectId, long typeId)	**long**	Return the number of tasks assigned to a user.

Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getAssignedTasksCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getActiveTasksCount

Description:

Method	Return values	Description
getActiveTasksCount(long statusId, long projectId, long typeId)	**Long**	Return the number of active tasks to a user.

Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getActiveTasksCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinishedTasksCount

Description:

Method	Return values	Description
getFinishedTasksCount(long statusId, long projectId, long typeId)	**Long**	Return the number of finished tasks to a user.

Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getFinishedTasksCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotifiedTasksCount

Description:

Method	Return values	Description
getNotifiedTasksCount(long statusId, long projectId, long typeId)	**Long**	Return the number of notified tasks assigned to a user.

Filter parameters description:

- The statusId parameters must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameters must be a valid type id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id
            long statusId = 1; // A valid status id
            long total = ws.task.getNotifiedTasksCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createTask

Description:

Method
createTask(String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> relatedDocume List<String> relatedFolders, List<String> relatedRecords, List<String> relatedMails, String repeatExpression, String repeatExpression, String formatDate, int repeatTimes, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)

The **repeatExpression** parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at l
of week) match the current time. The scheduler examines crontab entries once every minute. The time and date fiel

```
* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ????? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use na
? ? ? ?????????? month (1 - 12)
? ? ???????????????? day of month (1 - 31)
? ??????????????????? hour (0 - 23)
??????????????????????? min (0 - 59)
```

Example:

```
package com.openkm;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String subject = "task subject";

            Calendar calendar = Calendar.getInstance();
            String start = ISO8601.formatBasic(calendar);

            calendar.add(Calendar.DATE, 15);
            String end = ISO8601.formatBasic(calendar);

            String description = "description test";

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            List<String> notificationUsers = new ArrayList<>();
            notificationUsers.add("gnujavasergio");
            notificationUsers.add("sochoa");

            List<String> externalUsers = new ArrayList<>();
            externalUsers.add("gnu.java.sergio@gmail.com");
```

```
            List<String> relatedDocuments = new ArrayList<>();
            relatedDocuments.add("d701c503-87fc-4a9e-829e-478dc375eb83");
            relatedDocuments.add("f02053e3-2264-4040-bb84-67983a0208f7");

            List<String> relatedFolders = new ArrayList<>();
            relatedFolders.add("0a777b34-f518-4da7-9e58-8b1425e05add");
            relatedFolders.add("271cc3aa-218e-4574-8065-c34c704f4a20");

            List<String> relatedMails = new ArrayList<>();
            relatedMails.add("3ae1bb26-acdf-4463-a0c1-06cc62f55b95");
            relatedMails.add("c599be73-7831-4e93-be91-453baeda2b00");

            List<String> relatedRecords = new ArrayList<>();
            relatedRecords.add("96801b33-e92e-436b-8ed7-5d8dea904673");
            relatedRecords.add("23b50095-aa18-4d16-ae75-74b994a2f2b4");

            String currentDate = "31/04/2018";
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            calendar.setTime(sdf.parse(currentDate));

            String repeatExpression = "0 0 1 * *";

            calendar.add(Calendar.DATE, 1);
            String repeatUntil = ISO8601.formatBasic(calendar);

            // createTask
            Task newTask = ws.task.createTask(subject, start, end, description, statu
                    "okmAdmin", notificationUsers, externalUsers, relatedDocuments, r
                    "dd-MM-yyyy", 10, "m", 5, "m", 10);
            System.out.println(newTask);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateTask

Description:

Method
updateTask(long taskId, String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> relatedDocuments, List<String> relatedFolders, List<String> relatedRecords, List<String> relatedM String owner, String repeatExpression, String repeatUntil, String formatDate, int repeatTimes, int progress, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)

The **repeatExpression** parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at l of week) match the current time. The scheduler examines crontab entries once every minute. The time and date fiel

```
* * * * * command to execute
? ? ? ? ?
```

```
? ? ? ? ?
? ? ? ? ????? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use na
? ? ? ?????????? month (1 - 12)
? ? ?????????????? day of month (1 - 31)
? ????????????????????? hour (0 - 23)
??????????????????????????? min (0 - 59)
```

Example:

```
package com.openkm;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.util.ISO8601;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String subject = "update task subject";

            Calendar calendar = Calendar.getInstance();
            String start = ISO8601.formatBasic(calendar);

            calendar.add(Calendar.DATE, 15);
            String end = ISO8601.formatBasic(calendar);

            String description = "Update description test";

            long statusId = 1; // A valid task status id
            long projectId = 1; // A valid project id
            long typeId = 1; // A valid type id

            List<String> notificationUsers = new ArrayList<>();
            notificationUsers.add("gnujavasergio");
            notificationUsers.add("sochoa");

            List<String> externalUsers = new ArrayList<>();
            externalUsers.add("gnu.java.sergio@gmail.com");

            List<String> relatedDocuments = new ArrayList<>();
            relatedDocuments.add("d701c503-87fc-4a9e-829e-478dc375eb83");
            relatedDocuments.add("f02053e3-2264-4040-bb84-67983a0208f7");

            List<String> relatedFolders = new ArrayList<>();
            relatedFolders.add("0a777b34-f518-4da7-9e58-8b1425e05add");
            relatedFolders.add("271cc3aa-218e-4574-8065-c34c704f4a20");

            List<String> relatedMails = new ArrayList<>();
            relatedMails.add("3ae1bb26-acdf-4463-a0c1-06cc62f55b95");
```

```
            relatedMails.add("c599be73-7831-4e93-be91-453baeda2b00");

            List<String> relatedRecords = new ArrayList<>();
            relatedRecords.add("96801b33-e92e-436b-8ed7-5d8dea904673");
            relatedRecords.add("23b50095-aa18-4d16-ae75-74b994a2f2b4");

            String currentDate = "31/04/2018";
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            calendar.setTime(sdf.parse(currentDate));

            String repeatExpression = "0 0 1 * *";

            calendar.add(Calendar.DATE, 1);
            String repeatUntil = ISO8601.formatBasic(calendar);

            long taskId = 1;
            Task updateTask = ws.task.updateTask(taskId, subject, start, end, descrip
                    statusId, projectId, typeId, "okmAdmin", notificationUsers, exter
                    relatedDocuments, relatedFolders, relatedRecords, relatedMails,
                    "okmAdmin", repeatExpression, repeatUntil, "dd-MM-yyyy",
                    10, 10, "m", 5, "m", 10);
            System.out.println(updateTask);;
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTask

Description:

Method	Return values	Description
getTask(long taskId)	**Task**	Retrieve a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Task;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long taskId = 2;
            Task task = ws.task.getTask(taskId);
            System.out.println(task);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

deleteTask

Description:

Method	Return values	Description
deleteTask(long taskId)	**void**	Delete a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
     String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long taskId = 2;
            ws.task.deleteTask(taskId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createTaskProject

Description:

Method	Return values	Description
createTaskProject(String name, boolean active, String description)	**TaskProject**	Create a new task project.

The boolean parameter "active", indicates if your project is active or not.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            TaskProject tp = ws.task.createTaskProject("Project one", true, "Descript
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateTaskProject

Description:

Method	Return values	Description
updateTaskProject(long projectId, boolean active, String name, String description,)	**TaskProject**	Update a task project.

The boolean parameter "active", indicates if your project is active or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 4; // A valid project id
            TaskProject tp = ws.task.getTaskProject(projectId);
```

```
            System.out.println(tp);
            tp = ws.task.updateTaskProject(projectId, false, "cancelled", "Descriptio
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteTaskProject

Description:

Method	Return values	Description
deleteTaskProject(long projectId)	**void**	Delete a task project.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 5; // A valid project id
            ws.task.deleteTaskProject(projectId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskProject

Description:

Method	Return values	Description
getTaskProject(long projectId)	**TaskProject**	Return a task project object by id.

Example:

```
package com.openkm;
```

```
import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long projectId = 1; // A valid project id
            TaskProject tp = ws.task.getTaskProject(projectId);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskProjects

Description:

Method	Return values	Description
getTaskProjects(boolean filterActive)	**List<TaskProject>**	Retrieve a list of all the task projects.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskProject;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskProject tp : ws.task.getTaskProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createTaskType

Description:

Method	Return values	Description
createTaskType(String name, boolean active, String description)	**TaskType**	Create a new task type.

The boolean parameter "active", indicates if your type is active or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            TaskType tt = ws.task.createTaskType("Type one", true, "Description");
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateTaskType

Description:

Method	Return values	Description
updateTaskType(long typeId, boolean active, String name, String description)	**TaskType**	Update a task type.

The boolean parameter "active", indicates if your type is active or not.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long typeId = 4; // A valid type id
            TaskType tp = ws.task.getTaskType(typeId);
            System.out.println(tp);
            tp = ws.task.updateTaskType(typeId, false, "Type one", "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteTaskType

Description:

Method	Return values	Description
deleteTaskType(long typeId)	**void**	Delete a task type.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long typeId = 4; // A valid type id
            ws.task.deleteTaskType(typeId);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getTaskType

Description:

Method	Return values	Description
getTaskType(long typeId)	**TaskType**	Return a task type object by id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long typeId = 1; // A valid type id
            TaskType tt = ws.task.getTaskType(typeId);
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskTypes

Description:

Method	Return values	Description
getTaskTypes(boolean filterActive)	**List<TaskType>**	Retrieve a list of all the task types.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskType;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskType tt : ws.task.getTaskTypes(true)) {
                System.out.println(tt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createTaskStatus

Description:

Method	Return values	Description
createTaskStatus(String name, boolean finish)	**TaskStatus**	Create a new task status.

Depending your logic, several status can be ending status. For example status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            TaskStatus ts = ws.task.createTaskStatus("cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

updateTaskStatus

Description:

Method	Return values	Description
updateTaskStatus(long statusId, String name, boolean finish)	**TaskStatus**	Update a task status.

Depending your logic, several status can be ending status. For example status named "closed" or "cancelled". The boolean parameter "finish" indicates if your status in an "ending status".

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long statusId = 1; // A valid task id
            TaskStatus ts = ws.task.getTaskStatus(statusId);
            System.out.println(ts);
            ts = ws.task.updateTaskStatus(statusId, "cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteTaskStatus

Description:

Method	Return values	Description
deleteTaskStatus(long statusId)	**void**	Delete a task status.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long statusId = 4; // A valid status id
            ws.task.deleteTaskStatus(statusId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskStatus

Description:

Method	Return values	Description
getTaskStatus(long statusId)	**TaskStatus**	Return a task status object by id.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskStatus;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long statusId = 1; // A valid status id
            TaskStatus ts = ws.task.getTaskStatus(statusId);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskStatus

Description:

Method	Return values	Description
getTaskStatus()	**List<TaskStatus>**	Retrieve a list of all the task status.

Example:

```
package com.openkm;

import com.openkm.api.OKMTask;
import com.openkm.bean.TaskStatus;
import com.openkm.util.ContextWrapper;

public class Test {

    public static void main(String[] args) {
        try {
            OKMTask okmTask = ContextWrapper.getContext().getBean(OKMTask.class);
            for (TaskStatus ts : okmTask.getStatuses(null)) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createTaskNote

Description:

Method	Return values	Description
createTaskNote(long taskId, String text)	**TaskNote**	Create a new note to a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

```
        try {
            ws.login(user, password);
            long taskId = 1; // A valid task id
            TaskNote tn = ws.task.createTaskNote(taskId, "New note");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateTaskNote

Description:

Method	Return values	Description
updateTaskNote(long noteId, String text)	**TaskNote**	Update note of a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long noteId = 1; // A valid note id
            TaskNote tn = ws.task.updateTaskNote(noteId, "Note updated");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteTaskNote

Description:

Method	Return values	Description
deleteTaskNote(String token, long noteId)	**void**	Delete note of a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long noteId = 1; // A valid note id
            ws.task.deleteTaskNote(noteId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskNotes

Description:

Method	Return values	Description
getTaskNotes(long taskId)	**List<TaskNote>**	Retrieve a list of all the notes of a task.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.TaskNote;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long taskId = 1; // A valid task id
            for (TaskNote tn : ws.task.getTaskNotes(taskId)) {
                System.out.println(tn);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

UserConfig samples

Basics

Example of uuid:

- Using UUID -> "**373bcdd0-c082-4e7b-addd-e10ef813946e**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the UserConfig methods from "**userConfig**" class as is shown below:

```
ws.userConfig.getConfig();
```

Methods

getConfig

Description:

Method	Return values	Description
getConfig()	**UserConfig**	Returns the user settings.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.bean.UserConfig;
```

```
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            UserConfig userConfig = ws.userConfig.getConfig();
            Node node = ws.node.getNodeByUuid(userConfig.getHomeNode());
            System.out.println("User: " + userConfig.getUser());
            System.out.println("Home: " + node.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

setHome

Description:

Method	Return values	Description
setHome(String uuid)	**void**	Set the user home node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Node;
import com.openkm.sdk4j.bean.UserConfig;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.userConfig.setHome("53751cb7-c462-4320-ba46-4cc33e4667ae");

            UserConfig userConfig = ws.userConfig.getConfig();
            Node node = ws.node.getNodeByUuid(userConfig.getHomeNode());
            System.out.println("Home: " + node.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Wizard Samples

Basics

Example of uuid:

- Using UUID -> "**373bcdd0-c082-4e7b-addd-e10ef813946e**";

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Wizard methods from "**wizard**" class as is shown below:

```
ws.wizard.findByUser();
```

Methods

findByUser

Description:

Method	Return values	Description
findByUser()	**List<WizardNode>**	Returns a list of nodes with a wizard.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
import java.util.List;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            List<WizardNode> list = ws.wizard.findByUser();
            for (WizardNode wizardNode : list) {
                System.out.println(wizardNode);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByUserAndUuid

Description:

Method	Return values	Description
findByUserAndUuid(String uuid)	**WizardNode**	Get the wizard of a node

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.WizardNode;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            WizardNode wizardNode = ws.wizard.findByUserAndUuid("11225723-883f-4c12-8
            System.out.println(wizardNode);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasWizardByUserAndNode

Description:

Method	Return values	Description
hasWizardByUserAndNode()	**boolean**	Know if a node has a wizard

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            System.out.println("The node has a wizard: " + ws.wizard.hasWizardByUserA
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteAll

Description:

Method	Return values	Description
deleteAll()	**void**	Remove all wizards of the user.

Only wizards assigned to the user session will be removed.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.deleteAll();
            System.out.println("Delete all");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addDigitalSignature

Description:

Method	Return values	Description
addDigitalSignature()	**void**	Add digital signature in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.addDigitalSignature("055b5206-35d0-4351-ba92-c304bbba7ddf");
            System.out.println("addDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeDigitalSignature

Description:

Method	Return values	Description
removeDigitalSignature	**void**	Remove digital signature in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeDigitalSignature("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addShowWizardCategories

Description:

Method	Return values	Description
addShowWizardCategories	**void**	Add show wizard categories in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeDigitalSignature("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("removeDigitalSignature");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardCategories

Description:

Method	Return values	Description
removeShowWizardCategories	**void**	Remove show wizard categories in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeShowWizardCategories("02add6bf-e5ac-4f4a-8f3f-922958f5b6a
            System.out.println("removeShowWizardCategories");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addShowWizardKeywords

Description:

Method	Return values	Description
addShowWizardKeywords	**void**	Add show wizard keywords in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
```

```
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.addShowWizardKeywords("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("addShowWizardKeywords");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardKeywords

Description:

Method	Return values	Description
removeShowWizardKeywords	**void**	Remove show wizard keywords in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeShowWizardKeywords("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae"
            System.out.println("removeShowWizardKeywords");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addShowWizardOCRDataCapture

Description:

Method	Return values	Description
addShowWizardOCRDataCapture	**void**	Add show wizard OCR data capture in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.addShowWizardOCRDataCapture("02add6bf-e5ac-4f4a-8f3f-922958f5b6
            System.out.println("addShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardOCRDataCapture

Description:

Method	Return values	Description
removeShowWizardOCRDataCapture	**void**	Remove show wizard OCR data capture in the wizard of a node.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeShowWizardOCRDataCapture("02add6bf-e5ac-4f4a-8f3f-922958f
            System.out.println("removeShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addGroup

Description:

Method	Return values	Description
addGroup(String uuid, String group)	**void**	Add metadata group in the wizard.

The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.addGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl");
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeGroup

Description:

Method	Return values	Description
removeGroup(String uuid, String group)	**void**	Remove metadata group in the wizard.

The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeGroup("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "okg:tpl");
            System.out.println("removeGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addWorkflow

Description:

Method	Return values	Description
addWorkflow(String uuid, String workflow)	**void**	Add a workflow in the wizard.

The parameter **uuid** is the unique node identifier.

The parameter **workflow** is the workflow name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.addWorkflow("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "purchase")
```

```
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeWorkflow

Description:

Method	Return values	Description
removeWorkflow(String uuid, String workflow)	**void**	Remove a workflow in the wizard.

The parameter **uuid** is the unique node identifier.

The parameter **workflow** is the workflow name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.removeWorkflow("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae", "purchas
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setAutostart

Description:

Method	Return values	Description
setAutostart	**void**	Set auto start to the wizard.

Autostart should be always **set as the last wizard options**.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ws.wizard.setAutostart("02add6bf-e5ac-4f4a-8f3f-922958f5b6ae");
            System.out.println("setAutostart");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Workflow samples

Basics

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

Then should login using the method "**login**". You can access the "**login**" method from webservice object "**ws**" as is shown below:

```
ws.login(user, password);
```

Once you are logged with the webservices the session is keep in the webservice Object. Then you can use the other API method

At this point you can use all the Workflow methods from "**workflow**" class as is shown below:

```
ws.workflow.registerProcessDefinition(is);
```

For most examples it has been used the Purchase workflow sample.

Methods

registerProcessDefinition

Description:

Method	Return values	Description
registerProcessDefinition(InputStream is)	**void**	Registers a new workflow.

Example:

```
package com.openkm;

import java.io.FileInputStream;
import java.io.InputStream;

import org.apache.commons.io.IOUtils;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
```

```
public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            InputStream is = new FileInputStream("/home/openkm/Purchase.par");
            ws.workflow.registerProcessDefinition(is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteProcessDefinition

Description:

Method	Return values	Description
deleteProcessDefinition(long pdId)	**void**	Deletes a workflow.
The parameter pdId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long pdId = 1; // Valid workflow process definition
            ws.workflow.deleteProcessDefinition(pdId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessDefinition

Description:

Method	Return values	Description
getProcessDefinition(long pdId)	**void**	Returns a workflow process definition.
The parameter pdId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long pdId = 1; // Valid workflow process definition
            ProcessDefinition pd = ws.workflow.getProcessDefinition(pdId);
            System.out.println(pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

runProcessDefinition

Description:

Method	Return values	Description
runProcessDefinition(long pdId, String uuid, Map<String, String> propertiesMap)	**ProcessInstance**	Executes a workflow on a node.
The parameter pdId value is a valid workflow process definition. The parameter uuid can be any document, mail, folder or record UUID. The propertiesMap values are the form element values needed for starting the workflow (not all workflows need form values for starting).		

Example:

```
package com.openkm;

import java.util.HashMap;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long pdId = 8041; // Some valid workflow process definition id

            // Case as part of starting workflow are required form parameter
            Map<String, String> properties = new HashMap<>();
            properties.put("price", "1000");
            properties.put("description", "some description here");

            ws.workflow.runProcessDefinition(pdId, "f86cc22d-9b50-434f-a940-b04cea9c0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findProcessInstances

Description:

Method	Return values	Description
findProcessInstances(long pdId)	**List<ProcessInstance>**	Retrieves a list of all process instances of some registered workflows definition.
The parameter pdId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {
```

```
    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Get all workflow definitions
            for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
                System.out.println("WF definition: " + pd);

                // Get all process of some workflow definition
                for (ProcessInstance pi : ws.workflow.findProcessInstances(pd.getId()
                    System.out.println("PI: " + pi);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findAllProcessDefinitions

Description:

Method	Return values	Description
findAllProcessDefinitions()	**List<ProcessDefinition>**	Retrieves a list of all registered workflows definitions.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
                System.out.println(pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findLatestProcessDefinitions

Description:

Method	Return values	Description
findLatestProcessDefinitions()	**List<ProcessDefinition>**	Retrieves a list of the last workflows definitions.

Several versions of the same workflow can be registered.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Get all latest workflow definitions
            for (ProcessDefinition pd : ws.workflow.findLatestProcessDefinitions()) {
                System.out.println("WF definition: " + pd);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findLastProcessDefinition

Description:

Method	Return values	Description
findLastProcessDefinition(String name)	**ProcessDefinition**	Retrieves last workflow definition of some specific workflow.

The parameter name identifies an specific workflow definitions group.

Several workflow definition versions that have the same name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            ProcessDefinition pd = ws.workflow.findLastProcessDefinition("purchase");
            System.out.println("WF definition: " + pd);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessInstance

Description:

Method	Return values	Description
getProcessInstance(long piId)	**ProcessInstance**	Returns the process instance.
The parameter piId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long piId = 8108; // Some valid process instance id
```

```
            ProcessInstance pi = ws.workflow.getProcessInstance(piId);
            System.out.println("PI: " + pi);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findUserTaskInstances

Description:

Method	Return values	Description
findUserTaskInstances()	**TaskInstanceResultSet**	Retrieves a list of task instances assigned to the user.

Example:

```
package com.openkm.example;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.bean.workflow.TaskInstanceResultSet;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            TaskInstanceResultSet results = ws.workflow.findUserTaskInstances();
            System.out.println("Total: " + results.getTotal());
            for (TaskInstance ti : results.getResults()) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findTaskInstances

Description:

Method	Return values	Description
findTaskInstances(long piId)	**List<TaskInstance>**	Retrieves a list of task instances from a process instance id.
The parameter piId is a valid process instance id.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Get all task instances of some process instance
            long piId = 8108; // Some valid process instance id
            for (TaskInstance ti : ws.workflow.findTaskInstances(piId)) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTaskInstanceValues

Description:

Method	Return values	Description
setTaskInstanceValues(long tiId, String transName, Map<String, String> propertiesMap)	**void**	Set a task instance vales.

The parameter tiId is a valid task instance id.

The parameter transName is the choosen transaction.

The propertiesMap values are form element values needed for the workflow task (not all workflow tasks need form values).

Example:

```
package com.openkm;

import java.util.HashMap;
```

```
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long pdId = 2;
            long tiId = 8110; // Some valid task instance id
            TaskInstance ti = ws.workflow.getTaskInstance(tiId);
            Map<String, String> properties = new HashMap<>();
            ws.workflow.setTaskInstanceValues(pdId, ti.getId(), ti.getName(), "aprove
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTaskInstance

Description:

Method	Return values	Description
getTaskInstance(long tiId)	**TaskInstance**	Returns a task instance.
The parameter tiId is a valid task instance id.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long tiId = 8110; // Some valid task instance id
            TaskInstance ti = ws.workflow.getTaskInstance(tiId);
            System.out.println(ti);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

startTaskInstance

Description:

Method	Return values	Description
startTaskInstance(long tiId)	**void**	Starts a task instance.
The parameter tiId is a valid task instance id.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long tiId = 8110; // Some valid task instance id
            ws.workflow.startTaskInstance(tiId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTaskInstanceActorId

Description:

Method	Return values	Description
setTaskInstanceActorId(long tiId, String actorId)	**void**	Set the actor to a task instance.
The parameter tiId is a valid task instance id.		

The parameter actorId must be some valid OpenKM userId.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            long tiId = 8110; // Some valid task instance id
            ws.workflow.setTaskInstanceActorId(tiId, "okmAdmin");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

endTaskInstance

Description:

Method	Return values	Description
endTaskInstance(long tiId, String transName)	**void**	Ends a task instance.

The parameter tiId is a valid task instance id.

The parameter transName is a transaction name.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);
```

```
        try {
            ws.login(user, password);
            long tiId = 8110; // Some valid task instance id
            ws.workflow.endTaskInstance(tiId, "end");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessDefinitionForms

Description:

Method	Return values	Description
getProcessDefinitionForms(long pdId)	**Map<String, List<FormElement>>**	Return a map with all the process definition forms.
The parameter pdId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import java.util.List;
import java.util.Map;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.form.FormElement;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            Map<String, List<FormElement>> forms = ws.workflow.getProcessDefinitionFo
            for (String key : forms.keySet()) {
                System.out.println("Key:" + key);
                for (FormElement fe : forms.get(key)) {
                    System.out.println("Fe:" + fe);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProcessDefinitionImage

Description:

Method	Return values	Description
getProcessDefinitionImage(string pdId, string uuid)	**String**	Returns a workflow diagram in Base64.
The parameter pdId value is a valid workflow process definition.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            String tiId = "1";
            String res = ws.workflow.getProcessDefinitionImage(tiId, "82ed9618-11eb-4
            System.out.println(res);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findPooledTaskInstances

Description:

Method	Return values	Description
findPooledTaskInstances()	**TaskInstanceResultSet**	Retrieves a list of all pooled task instances.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.TaskInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
```

```
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            for (TaskInstance ti : ws.workflow.findPooledTaskInstances()) {
                System.out.println(ti);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findProcessInstancesByNode

Description:

Method	Return values	Description
findProcessInstancesByNode(String uuid)	**List<ProcessInstance>**	Retrieves a list of all process instances by node of some registered workflows definition.
The parameter uuid can be any document, mail, folder or record UUID.		

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.workflow.ProcessDefinition;
import com.openkm.sdk4j.bean.workflow.ProcessInstance;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);
            // Get all workflow definitions
            for (ProcessDefinition pd : ws.workflow.findAllProcessDefinitions()) {
                System.out.println("WF definition: " + pd);

                // Get all process of some workflow definition
                for (ProcessInstance pi : ws.workflow.findProcessInstances(pd.getId()
                    System.out.println("PI: " + pi);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

getSuggestBoxKeyValue

Description:

Method	Return values	Description
getSuggestBoxKeyValue(long pdId, String uuid, String taskName, String propertyName, String key)	**String**	Returns the suggestBox value for a key.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            long pdId = 4;
            String taskName = "task-elaboration";
            System.out.println(ws.workflow.getSuggestBoxKeyValue(pdId, "301de803-f4ae
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValuesFiltered

Description:

Method	Return values	Description
getSuggestBoxKeyValuesFiltered(long pdId, String uuid, String taskName, String propertyName, String filter)	**Map<String, String>**	Retrieves a map - (key, value) pairs - with suggestBox values

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;

import java.util.Map;

public class Test {

    public static void main(String[] args) {
        String host = "http://localhost:8080/openkm";
        String user = "okmAdmin";
        String password = "admin";
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.login(user, password);

            Map<String, String> values = ws.workflow.getSuggestBoxKeyValuesFiltered(4
            for (String key : values.keySet()) {
                String value = values.get(key);
                System.out.println(key + ":" + value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

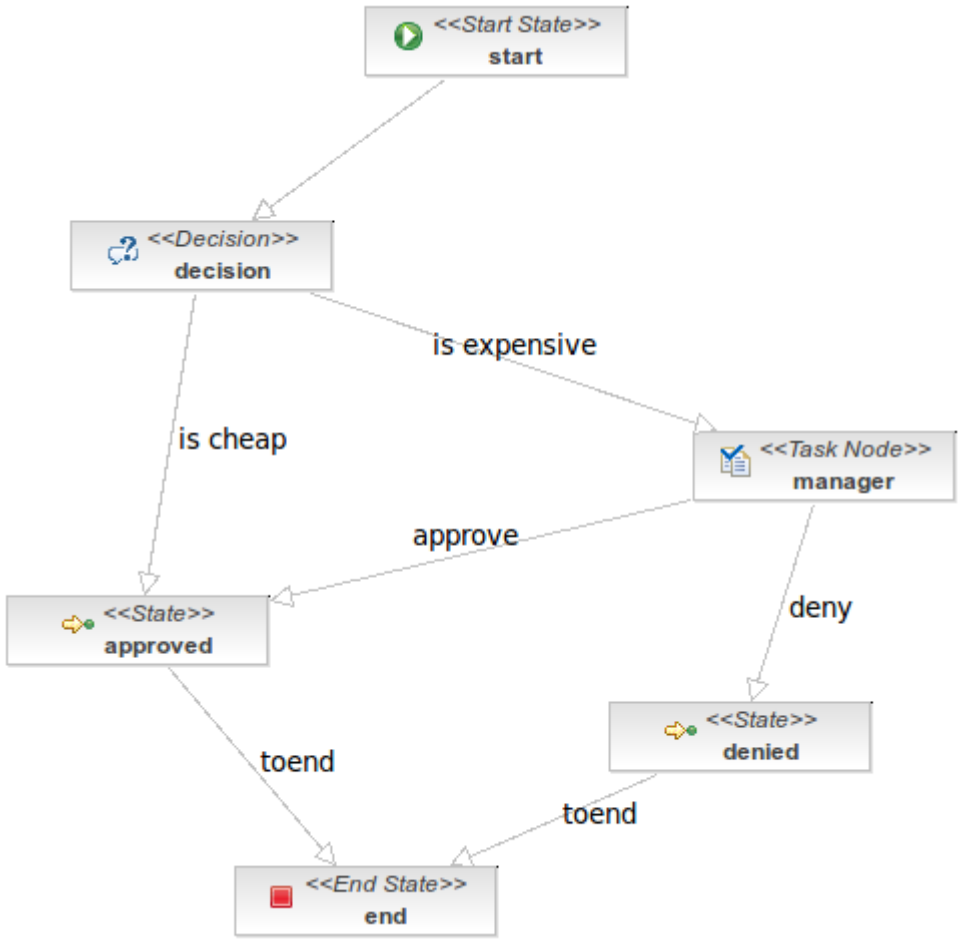
Purchase workflow sample

The tasks will be assigned to an user called "manager" so you need to create this user and log as such to see the task assignment.

Also you can assign this task to another user from the process instance workflow administration.

Download the Purchase.par file.

Process image



Process definition

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="purchase">
  <start-state name="start">
    <transition to="decision"></transition>
  </start-state>

  <decision name="decision">
    <transition to="approved" name="is cheap">
      <condition expression="#{price.value &lt;= 500}"></condition>
```

```
    </transition>
    <transition to="manager" name="is expensive">
      <condition expression="#{price.value &gt; 500}"></condition>
    </transition>
  </decision>

  <task-node name="manager">
    <task name="evaluate price">
      <description>The manager may deny purchase or go ahead.</description>
      <assignment actor-id="manager"></assignment>
    </task>
    <transition to="denied" name="deny"></transition>
    <transition to="approved" name="approve"></transition>
  </task-node>

  <state name="approved">
    <description>The purchase has been approved.</description>
    <timer duedate="15 seconds" name="approved timer" transition="toend">
      <script>print(&quot;From APPROVED Go to END&quot;);</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <state name="denied">
    <description>The purchase has been denied.</description>
    <timer duedate="15 seconds" name="denied timer" transition="toend">
      <script>print(&quot;From DENIED Go to END&quot;);</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <end-state name="end"></end-state>
</process-definition>
```

Process handlers

None.

Form definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
                                "http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="run_config">
    <input label="Purchase price" name="price" />
    <textarea label="Purchase description" name="description" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="evaluate price">
    <input label="Purchase price" name="price" data="price" readonly="true" />
    <textarea label="Purchase description" name="description" data="description" read
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>
```