Documentation for SDK for Java 5.4

Table of Contents

SDK for Java 5.4

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 web services library.

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

License

SDK for Java is licensed under the terms of the EULA - OpenKM SDK End User License Agreement and is 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

The SDK for Java version 5.4 should be used:

- **For OpenKM Professional version 8.2.4 or later**

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.org/xsd/maven-4.0.0.xsd">
  <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>5.4</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.getChildren("4f873d10-654e-4d99-a94f-15466e30a0f6")) {
                System.out.println("Folder -> " + fld.getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Jar sample

If you use "**maven-assembly-plugin**" rather than "**maven-shade-plugin**" you will get an error "**missing MessageBodyWriter**" while uploading a new file via 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/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

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

  <properties>
    <java.compiler>17</java.compiler>
    <sdk4j.version>5.4</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.ManifestResourceTransformer"
```

```
              <mainClass>com.openkm.Main</mainClass>
            </transformer>
            <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
              <resource>META-INF/services/javax.ws.rs.ext.MessageBodyWriter</resource>
            </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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

At this point you can use all the Repository methods from the "**repository**" class as 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 shown in this Javadoc API summary.

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

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

There is a direct correspondence between the classes and methods implemented in the com.openkm.api packages and available from the 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 get the profiles.	Yes	AuthImpl.java	BaseAuth.java
OKMBookmark	Manages the user bookmarks.	Yes	BookmarkImpl.java	BaseBookmark.java
OKMDashboard	Manages all data shown on the 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 mail. For example, create, move or delete a mail.	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 a metadata group or set	Yes	PropertyGroupImpl.java	BasePropertyGroup.java

	metadata fields.			
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	Many options related to the repository. For example, get the properties of the main root node (/okm:root).	Yes	RepositoryImpl.java	BaseRepository.java
OKMReport	Manages reports. For example, execute reports.	Yes	ReportImpl.java	BaseReport.java
OKMSearch	Manages the search feature. For example, manage saved queries or perform a new query on the repository.	Yes	SearchImpl.java	BaseSearch.java
OKMStamp	Manages stamps.	Yes	StampImpl.java	BaseStamp.java
OKMStats	General repository statistics.	No		

OKMTask	Manages tasks. For example, create a new task.	Yes	TaskImpl.java	BaseTask.java
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** returns an **com.openkm.OKMWebservices** object which implements all interfaces. `OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);`
com.openkm.sdk4j.bean	Contains all classes resulting from 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 you should log in using the "**login**" method. You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

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

Methods

findActivityLog

Description:

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

Example:

```
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, endDate, user, "", item);
            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 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 change or get security grants.

To set READ and WRITE access, do the following:

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

To check if you have permission access, do the following:

```
// 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 web service object:

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

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

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

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

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

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

Methods

login

Description:

Method	Return values	Description
login(String user, String password)	String	The login process returns an authentication token.

By default, the login token has an **expiration time of 24 hours**.

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

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

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

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

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

Example:

```
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	The login process returns an authentication token.

By default, the login token has an **expiration** time determined by the **expiration value in days**.

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

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

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

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

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

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

Example:

```
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();
        }
    }
}
```

setAuthorizationToken

Description:

Method	Return values	Description

setAuthorizationToken(String authorizationToken)	void	Authentication process using an authorization token.

For applications that require persistent integration with the API, it is good practice to generate a long-lived token (using the login method) and then reuse it. This way, the integration does not require a constant login process and remains logged in.

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";
        // Usually a long-term authorization token
        String authorizationToken = "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqbGxvcnQiLYRpsCI6IiIsInJvbGUiOlsiUk9MF
        OKMWebservices ws = OKMWebservicesFactory.getInstance(host);

        try {
            ws.setAuthorizationToken(authorizationToken);
            // at this point, you have grants to request the API
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

logout

Description:

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

The 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();
        }
    }
}
```

getRefreshToken

Description:

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

When logging in, an authentication token is set in the OKMWebservice. By default, the authentication token expires in 24 hours. This method allows replacing the current token with a fresh one.

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);
            // getRefreshToken
            String token = ws.getRefreshToken();
            System.out.println("New Token: " + token);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGrantedUsersAndRoles

Description:

Method	Return values	Description
getGrantedUsersAndRoles(String uuid)	**GrantedUsersAndRolesItem**	Returns 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("3c68b3a1-c65c-4b1e-84b5-9ce27
            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>**	Returns 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-addd-e10ef813946e");
            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>**	Returns 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-addd-e10ef813946e");
            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>**	Returns the list of all roles.
When the showAll variable is set to true, it returns all the roles â€” enabled and disabled; otherwise it returns only the enabled ones.		

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>**	Returns the list of all 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>**	Returns the list of all users.
When the showAll variable is set to true, it returns all the users â€” enabled and disabled; otherwise it returns only the enabled ones.		

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**	Returns 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>**	Returns the list of all users who have been 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**	Removes a role grant on a node.
The 'recursive' parameter only applies when the uuid is a folder or a record node. When the recursive parameter 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", Permission.ALL_GRANTS

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

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

revokeUser

Description:

Method	Return values	Description
revokeUser(String uuid, String user, int permissions, boolean recursive)	**void**	Removes a user grant on a node.
The 'recursive' parameter only applies when the uuid is a folder or a record node. When the recursive parameter 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 sochoa write grants at the node but not descendants
            ws.auth.revokeUser("373bcdd0-c082-4e7b-addd-e10ef813946e", "sochoa", Permission.ALL_GRANTS, false

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

grantRole

Description:

Method	Return values	Description

grantRole(String uuid, String role, int permissions, boolean recursive)	**void**	Adds a role grant on a node.
The 'recursive' parameter only applies when the uuid is a folder or a record node. When the recursive parameter 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);
            // Add ROLE_USER write grants at the node but not descendants
            ws.auth.grantRole("4f873d10-654e-4d99-a94f-15466e30a0f6", "ROLE_USER", Permission.ALL_GRANTS, f

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

grantUser

Description:

Method	Return values	Description
grantUser(String uuid, String user, int permissions, boolean recursive)	**void**	Adds a user grant on a node.
The 'recursive' parameter only applies when the uuid is a folder or a record node. When the recursive parameter 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);
            // Add sochoa write grants at the node but not descendants
            ws.auth.grantUser("4f873d10-654e-4d99-a94f-15466e30a0f6", "sochoa", Permission.ALL_GRANTS, false);

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

changeSecurity

Description:

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

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

Although values set in the revokeUsers and revokeRoles variables of the ChangeSecurity object may be provided, these values will not be taken into consideration when overwriting 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();
        }
    }
}
```

getSessionId

Description:

Method	Return values	Description
getSessionId()	**String**	Gets the 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", true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteUser

Description:

Method	Return values	Description
deleteUser(String userId)	**void**	Deletes 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**	Updates 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**	Deletes 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**	Updates 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**	Assigns a 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**	Removes 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>**	Returns the list of all profiles.

When the filterByActive parameter is enabled, the method will return only the active profiles; otherwise it will return all available profiles.

Each user is assigned one profile that enables more or fewer 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**	Returns the profile assigned to a user.

Each user is assigned one profile that enables more or fewer 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 for 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**	Returns the token for 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**	Updates user permissions on a node.
The 'recursive' parameter only applies when the uuid is a folder or a record node.		

When the recursive parameter 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);
            // Update permissions of sochoa at the node but not descendants
            ws.auth.setUserPermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "sochoa", Permission.READ + Pe

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

setRolePermissions

Description:

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

The 'recursive' parameter only applies when the uuid is a folder or a record node.

When the recursive parameter 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);
            // Update permissions of ROLE_USER at the node but not descendants
            ws.auth.setRolePermissions("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573", "ROLE_USER", Permission.REAI

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

getUserTenants

Description:

Method	Return values	Description
getUserTenants()	**List<Tenant>**	Returns 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 for a user.

A 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**	Returns true when the user's login must be set in lowercase.

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();
        }
    }
}
```

getToken

Description:

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

The authentication token expires after 24 hours or later (it depends on how it was created). This method helps refresh 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();
        }
    }
}
```

createPersonalAccessToken

Description:

Method	Return values	Description
createPersonalAccessToken(String name, Calendar expiration)	void	Create a personal access authentication token

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Calendar;

public class Test {

    private static final Logger log = LoggerFactory.getLogger(Test.class);

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

        try {
            ws.login(user, password);
            // createPersonalAccessToken
            Calendar expiration = Calendar.getInstance();
            expiration.set(Calendar.DAY_OF_MONTH, +10);
            String personalToken = ws.auth.createPersonalAccessToken("openkm", expiration);
            log.info(personalToken);
        } catch (Exception e) {
            log.error(e.getMessage());
        }
    }
}
```

Bookmark samples

Basics

Suggested code sample

First, you must create the web service object:

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

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

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

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

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

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

Methods

getUserBookmarks

Description:

Method	Return values	Description
getUserBookmarks()	**List<Bookmark>**	Returns a list of all bookmarks for 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();
        }
    }
}
```

create

Description:

Method	Return values	Description
create(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.create("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "test bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(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.rename(bookmarkId, "set bookmark");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(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.delete(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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

At this point you can use all the Conversion methods from "**conversion**" class as 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. The application uses this parameter to identify the document MIME TYPE by its extension.		
 The OpenOffice service must be enabled on the 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 actually executing the ImageMagick convert tool on the server side.

You can set many parameters (transformations) in the **params** variable. Take a look at ImageMagick convert tool to get a complete list of them.

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

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

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

When building your integrations, we suggest installing ImageMagick software locally, and checking 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", "-resize 50% ${fileIn} ${fileOut}"
            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 a URL.

The HTML to PDF conversion tool must be enabled on the 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 uploaded document.

A Text Extractor must be enabled for the document MIME TYPE on the 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 on the 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 on the 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 you should log in using the "**login**" method. You can access the "**login**" method from the web service object "**ws**" as shown below:

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

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

At this point you can use all the Dashboard methods from "**dashboard**" class as 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 being edited by the user.

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

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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();
        }
    }
```

```
}
```

getUserLockedFolders

Description:

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

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

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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.getUserLastDocumentsNotesCreated(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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.getUserLastFoldersNotesCreated(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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.getUserLastRecordsNotesCreated(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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.getUserLastDownloadedDocuments(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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(0, 5);
            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 parameters "limit" and "offset" allow you to retrieve only a portion of the results of a query.

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

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

- limit=10
- offset=0

Now, suppose you want to show the results from 11-20; 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.getUserLastImportedMailAttachments(0, 5);
            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 on 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 a 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 in with the web services, the session is kept in the web service object. Then you can use the other API methods.

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

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

Methods

create

Description:

Method	Return values	Description
create(String uuid, String name, InputStream is)	**Document**	Creates a new document. Returns a Document object with the properties of the created document.
The value 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/openkm/okm/logo.png");
            Document doc = ws.document.create("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is);
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

create

Description:

Method	Return values	Description
create(String uuid, String name, InputStream is, long nodeClass)	**Document**	Creates a new document with a nodeClass. Returns a Document object with the properties of the created document.
The value of the uuid parameter should be a folder or record node UUID. The nodeClass parameter should be a valid business type (series) value. A nodeClass with value 0 means there is no business type (series) 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/openkm/okm/logo.png");
            long nodeClass = 0;
            Document doc = ws.document.create("1be884f4-5758-4985-94d1-f18bfe004db8", "logo.png", is, nodeClass)
            System.out.println(doc);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

create

Description:

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

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/openkm/okm/logo.png");
            Document doc = ws.document.create("151f3a54-f370-47d6-801a-d20faecec180", file);
            System.out.println(doc);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

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

When a document is deleted, it is automatically moved to the /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.delete("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	Return values	Description
getProperties(String uuid)	**Document**	Returns 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.getProperties("1ec49da9-1746-4875-ae32-9281d7303a62"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	Return values	Description
getContent(String uuid)	**InputStream**	Retrieves the document content - binary data - of the current document version.

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

```
Document doc = ws.getDocumentProperties("1ec49da9-1746-4875-ae32-9281d7303a62");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

We've encountered incorrect size problems when 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-ae32-9281d7303a62");
            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 document content (binary data) from a specific document version.

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

```
Document doc = ws.getProperties("1ec49da9-1746-4875-ae32-9281d7303a62");
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

We've encountered incorrect size problems when 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", "1.1");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String fldId)	**List<Document>**	Returns a list of all documents whose parent is fldId.
The parameter fldId can be a folder or a record node 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.getChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(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.rename("1ec49da9-1746-4875-ae32-9281d7303a62", "logo_rename.png");
            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 title", "some description", "es
        } 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();
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(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.setTitle("1ec49da9-1746-4875-ae32-9281d7303a62", "Some title here");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkout

Description:

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

Only one user can modify a document at a time.

Before starting editing, you must perform a checkout action that locks the editing process for other users and allows editing only to the user who 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 user who executed the action
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

cancelCheckout

Description:

Method	Return values	Description
cancelCheckout(String uuid)	**void**	Cancels document editing.

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 user who executed the action
            ws.document.cancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

forceCancelCheckout

Description:

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

This method allows cancelling editing on any document.

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

This action can only be performed by a superuser (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 user who executed the action
            ws.document.forceCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isCheckedOut

Description:

Method	Return values	Description
isCheckedOut(String docId)	**Boolean**	Returns a boolean that indicates whether the document is being edited 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("1ec49da9-1746-4875-ae32-9
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

checkin

Description:

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

String comment)		with new Version values.

Only the user who started the editing (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 some comment");
            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 the increment variable must be 1 or greater.

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

Only the user who started the editing (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 some comment", 2);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

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

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

When a document is purged, it can only be restored from a previous repository backup. The purge action permanently removes the document 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.purge("1ec49da9-1746-4875-ae32-9281d7303a62");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

move

Description:

Method	Return values	Description
move(String uuid, String dstId)	**void**	Moves a document 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.document.move("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e", "8599eab7-ae61-4628-8010-1103d6950c63
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	Return values	Description
copy(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 preserve the same name.

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

See "**extendedCopy**" 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", "8599eab7-ae61-4628-8010-1103
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getVersionHistorySize

Description:

Method	Return values	Description
getVersionHistorySize(String uuid)	**long**	Returns the total size in bytes of all documents in the document 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-fd8e2b47f55e");
            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, UNITS[i]);
                }
            }

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

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

isValid

Description:

Method	Return values	Description
isValid(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.isValid("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPath

Description:

Method	Return values	Description
getPath(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.getPath("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDetectedLanguages

Description:

Method	Return values	Description
getDetectedLanguages()	**List<String>**	Returns a list of available document languages that 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();
        }
    }
}
```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(String nodeId, String dstId, String name, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security)	**void**	Copies a document with associated data into some 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 are copied; 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 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.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.extendedCopy("055b5206-35d0-4351-ba92-c304bbba7ddf", "2dccdee1
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getExtractedText

Description:

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

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

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

Despite this restriction, via the API only administrator users can perform database queries to retrieve this information. Check Repository samples for accessing database-level information.

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**	Sets the 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-ad3017dd78a7");
            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-ad3017dd78a7", ThumbnailType.T
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

createFromTemplate

Description:

Method	Return values	Description
createFromTemplate(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, these values are mandatory and must be set in the properties parameter.

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

Additional:

- When the category parameter is true, the original values of the categories will be copied.
- When the keywords parameter is true, the original values of the keywords will be copied.
- When 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.createFromTemplate("5e72dec8-2409-405b-ac15-d964feb3d7bb", "/ok
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateFromTemplate

Description:

Method	Return values	Description
updateFromTemplate(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 UUID of the document to be updated.

This method only makes 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.updateFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "058b9379-b441-454d-8590
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAnnotations

Description:

Method	Return values	Description
getAnnotations(String uuid, String versionName)	**String**	Returns the document annotations for a given 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-fd8e2b47f55e", "1.0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getDifferences

Description:

Method	Return values	Description
getDifferences(String uuid, String versionName1, String versionName2)	**InputStream**	Returns 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-ad3017dd78a7", "1.0", "1.1");
            FileOutputStream fos = new FileOutputStream("/home/openkm/okm/out.pdf");
            IOUtils.copy(is, fos);
            is.close();
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getCheckedOut

Description:

Method	Return values	Description
getCheckedOut()	**List<Document>**	Returns a list of documents checked out 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();
        }
    }
}
```

setNodeClass

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.setNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(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.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(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.setDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

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

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

The WizardNode contains a list of pending actions that 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/openkm/okm/logo.png");
            WizardNode wn = ws.document.createWizard("1f323e88-64ee-4f57-91e2-9276f8c603f9", "logo.png", 0, is);
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isOCRDataCaptureSupported

Description:

Method	Return values	Description
isOCRDataCaptureSupported(String uuid)	**boolean**	Checks if the node supports 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-abf76f9ec499")) {
                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**	Returns 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-abf76f9ec499");
            System.out.println(ocr);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

captureData

Description:

Method	Return values	Description
captureData(String uuid, long templateId)	**void**	Proceed with the OCR data capture using the 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", templateId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNumberOfPages

Description:

Method	Return values	Description
getNumberOfPages(String uuid)	**int**	Gets 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.getNumberOfPages("b2f88679-e3fd-4f97-bf0
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPageAsImage

Description:

Method	Return values	Description
getPageAsImage(String uuid, int pageNumber, int	**String**	Returns a specific page as an image encoded

maxWidth, int maxHeight)		as a Base64 string.

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-6ac4ba6c1524", 1, 150, 150)
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

liveEditCheckin

Description:

Method	Return values	Description
liveEditCheckin(String uuid, String comment, int increment)	**void**	Performs a live edit check-in.

The method should be used only when the document has been edited using 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", "comment 1.1", 1);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

liveEditCancelCheckout

Description:

Method	Return values	Description
liveEditcancelCheckout(String uuid)	**void**	Cancels live editing of a document.

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 user who executed the action
            ws.document.liveEditCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

liveEditForceCancelCheckout

Description:

Method	Return values	Description
liveEditForceCancelCheckout(String uuid)	**void**	Cancels live editing of a document.

This method allows canceling live editing on any document.

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

This action can only be done by a superuser (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 user who executed the action
            ws.document.liveEditForceCancelCheckout("1ec49da9-1746-4875-ae32-9281d7303a62");
            // At this point other users are allowed to execute a checkout and modify the document
        } 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**	Updates the content of the document edited with the 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 whether the node is a mail attachment.

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-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isConvertibleToPDF

Description:

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

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

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("9ed5c7b1-9314-4479-8f80-fd8e2b47f55e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPdf

Description:

Method	Return values	Description
getPdf(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.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            InputStream is = ws.document.getPdf(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();
        }
    }
}
```

saveAsPdf

Description:

Method	Return values	Description
saveAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)	**Document**	Saves the document as a PDF in the OpenKM repository.

The value of the **uuid** parameter should be a Document UUID.

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

Additional:

- When the propertyGroups parameter is true, the metadata groups of the source are copied to the target.
- When the security parameter is true, the security of the source is copied to the target.

Example:

```
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.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            Document docPdf = ws.document.saveAsPdf(doc.getUuid(), "book.pdf", true, true);
            System.out.println("Document pdf: " + docPdf.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

webPageImport

Description:

Method	Return values	Description
webPageImport(String uuid, String url)	**void**	Saves a document as a PDF with the content of the web page.
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://www.google.com/");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setIndexable

Description:

Method	Return values	Description

setIndexable(String uuid, boolean enabled)	**void**	Sets whether the document is 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-90c8-6b435e4b7d79");
            System.out.println("Document indexable:" + document.isIndexable());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getXRefs

Description:

Method	Return values	Description
getXRefs(String uuid)	**List<XRef>**	Returns a list of the document references of an 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.document.getXrefs("5ce6dd37-7472-404a-878e-274005a2d6db")) {
                System.out.println(xRef);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

refresh

Description:

Method	Return values	Description
refresh(String uuid)	**void**	Refreshes the 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.document.refresh("5ce6dd37-7472-404a-878e-274005a2d6db");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

FilePlan samples

Basics

Suggested code sample

First, you must create the web service object:

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

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

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

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

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

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

Methods

getRootSections

Description:

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

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 NodeClassSectionDefinition objects filtered by permissions.

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get 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.getRootSectionsFilteredBySecurity(Permission.ALL_GR
            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 NodeClassSectionDefinition objects for the 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(parentId);
            for (NodeClassSectionDefinition nodeClassSectionDefinition : list) {
                System.out.println(nodeClassSectionDefinition);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildrenSectionsFilteredBySecurity

Description:

Method	Return values	Description
getChildrenSectionsFilteredBySecurity(long	**List<NodeClassSectionDefinition>**	Returns a list of NodeClassSectionDefinition

parentId, int permissions)		objects for the parent section and filtered by permissions.

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get 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.getChildrenSectionsFilteredBySecurity(parentId, Permiss
            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 NodeClass objects for the 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 NodeClass objects for the section and filtered by permissions.

The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get 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(sectionId, Permission.WRITE);
            for (NodeClass nodeClass : list) {
                System.out.println(nodeClass);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findNodeClassByPk

Description:

Method	Return values	Description
findNodeClassByPk(long id)	**NodeClass**	Returns 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 NodeClass objects that are Electronic 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 NodeClass objects that are Electronic Records for the section and filtered by permissions.

> The class **com.openkm.sdk4j.bean.Permission** contains permission values (READ, WRITE, etc.). You should use it in combination with methods that change or get 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.findElectronicRecordClassesFilteredBySecurity(Permission.ALL_GRANTS
            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 change or get 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.findFilteredByCodeOrNameFilteredBySecurity("1.", "", Permission.ALL_G
            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 change or get 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("1.", "invoice", Permission.ALL_GRAI
            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 NodeClassSectionDefinition objects that 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(sectionId);
            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 NodeClass objects.

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 section IDs that have children and whose 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 section IDs that have children with a parent equal 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(parentId);
            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 web service object:

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

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

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

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

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

```
ws.folder.create("1be884f4-5758-4985-94d1-f18bfe004db8", "test");
```

Methods

create

Description:

Method	Return values	Description
create(String uuid, String name)	**Folder**	Creates a new folder and returns a Folder object.
The parameter **uuid** should be a 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.create("1be884f4-5758-4985-94d1-f18bfe004db8", "test");
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	Return values	Description
getProperties(String uuid)	**Folder**	Returns 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.getProperties("76f4155e-42af-4be4-9a1c-756497616fb6"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(String fldId)	**void**	Deletes 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.delete("76f4155e-42af-4be4-9a1c-756497616fb6");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(String fldId, String newName)	**void**	Renames 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.rename("91264771-02b9-471d-b7a3-ba8cc49a10dd", "renamedFolder");
```

```
            // Folder has renamed to /okm:root/renamedFolder
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

move

Description:

Method	Return values	Description
move(String uuid, String dstId)	**void**	Moves a folder into another 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.move("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8010-1103d6950c63");
            // Folder has moved to /okm:root/tmp/test
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String uuid)	**List<Folder>**	Returns a list of all folders whose 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.getChildren("1be884f4-5758-4985-94d1-f18bfe004db8")) {
                System.out.println(fld);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	Return values	Description
isValid(String uuid)	**Boolean**	Returns a boolean indicating whether the node is 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);
            // Return false
            ws.folder.isValid("8cd1e072-8595-4dd3-b121-41d622c43f08");

            // Return true
            ws.folder.isValid("1be884f4-5758-4985-94d1-f18bfe004db8");
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getPath

Description:

Method	Return values	Description
getPath(String uuid)	**String**	Converts a folder UUID to a 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.getPath("1be884f4-5758-4985-94d1-f18bfe004db8"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	Return values	Description
copy(String uuid, String dstId, String newName)	**void**	Copies a folder into another folder or record.

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

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

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

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

Example:

```
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.copy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8010-1103d6950c63", "n
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(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 are copied; the metadata, keywords, etc. of the folder are not copied.

Additional:

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

- When the notes parameter is true, the original values of the notes will be copied.
- When the security parameter is true, the original value of the security will be copied.

Example:

```
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.extendedCopy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628
                    "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**	Returns a ContentInfo object with information about the folder.

The ContentInfo object retrieves information about:

- The number of folders inside the folder.
- The number of documents inside the folder.
- The number of records inside the folder.
- The number of mails inside the folder.
- The size in bytes of all objects in 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-bdc1-74181cafbc5d"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	Return values	Description
purge(String uuid)	**void**	The folder is permanently removed from the repository.

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

When a folder is purged, it will only be able to be restored from a previous repository backup. The purge action removes the folder permanently 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.purge("ada67d44-b081-4b23-bdc1-74181cafbc5d");
```

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

setStyle

Description:

Method	Return values	Description
setStyle(String uuid, long styleId)	**void**	Sets 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**	Creates 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/missingfld3");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(String uuid, String description)	**void**	Sets 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.setDescription("4c195453-246b-4ce9-86ba-b84e68d1f284", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFromTemplate

Description:

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

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, these values are mandatory and must be set in the properties parameter.

Additional:

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

Example:

```
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.createFromTemplate("47c70047-2924-483c-bc42-23d0cbef6b17", "/okm:root/sdk/te
            System.out.println(folder);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Import samples

Basics

We suggest executing the methods below for large imports or for simple creation cases.

These methods execute fewer steps in the background than what is described in Document samples and Folder samples. That means you will get better performance because less logic is executed on the server side.

Suggested code sample

First, you must create the webservice object:

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

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

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

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

At this point you can use all the import methods from "**quickImport**" class as 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. Returns the UUID of the document.
The value 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-f18bfe004db8", file);
            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. Returns the UUID of the folder.
The value 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-f18bfe004db8", "folder-name");
            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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

```
ws.mail.getProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbced")
```

Methods

getProperties

Description:

Method	Return values	Description
getProperties(String uuid)	**Mail**	Returns the mail properties.

Example:

```
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.getProperties("05d9b8e3-f9c1-4ace-9007-afd775dbbced"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(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.delete("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	Return values	Description
purge(String uuid)	**void**	Mail is permanently removed from the repository.

Usually, you will purge mails to /okm:trash/userId - the user's personal trash location - but it is possible to directly purge any mail from the entire repository.

When a mail is purged, it can only be restored from a previous repository backup. The purge action permanently removes the mail 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.purge("05d9b8e3-f9c1-4ace-9007-afd775dbbced");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(String uuid, String newName)	**void**	Renames a mail.

In the OpenKM frontend UI, the subject is used to display the mail name in the file browser table. That means this change will take effect internally on the mail path, but will not be reflected in the 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.rename("e3831cc4-30f0-419a-8fbf-a3418472ada5", "new name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

move

Description:

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

Example:

```
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.move("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafbc5d");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Return

Method	values	Description
public void copy(String uuid, String dstId, String newName)	**void**	Copies a mail into a folder or record.

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

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

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

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

Example:

```
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.copy("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-74181cafbc5d", "new
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(String uuid, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean security, String newName)	**Mail**	Copies a mail with associated data into a folder or record.

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

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

Additional:

- When the categories parameter is true, the original values of the categories will be copied.
- When the keywords parameter is true, the original values of the keywords will be copied.
- When the 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 security grants will be copied.
- When newName is set, the mail's name is changed.

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.extendedCopy("e3831cc4-30f0-419a-8fbf-a3418472ada5", "ada67d44-b081-4b23-bdc1-
            System.out.println(mail);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String uuid)	**List<Mail>**	Returns a list of all mails whose 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.getChildren("ada67d44-b081-4b23-bdc1-74181cafbc5d")) {
                System.out.println(mail);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	Return values	Description
isValid(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.isValid("ada67d44-b081-4b23-bdc1-74181cafbc5d"));

            // Return true
            System.out.println(ws.mail.isValid("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
```

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

getPath

Description:

Method	Return values	Description
getPath(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.getPath("e3831cc4-30f0-419a-8fbf-a3418472ada5"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createAttachment

Description:

Method	Return values	Description
createAttachment(String uuid, String docName, InputStream is)	**Document**	Adds an attachment 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.png", is);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteAttachment

Description:

Method	Return values	Description
deleteAttachment(String uuid, String docId)	**void**	Deletes a mail attachment.
The value of the **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", "6dcb02dc-0881-4610-b2bc-a158ed7
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAttachments

Description:

Method	Return values	Description
getAttachments(String uuid)	**List<Document>**	Retrieves a list of all attachment documents of the mail.

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-a3418472ada5")) {
                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**	Sends a mail message with attachments.

The value 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 it is sent.

Other parameters:

- to is a list of destination email accounts.
- subject is the mail subject.
- cc, bcc, replyTo are lists of destination email accounts (optional)
- docsId is a list of valid document UUIDs already in OpenKM that will be sent as attachments in the 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", "8cd1e072-8595-4dd3-b12
            ws.mail.sendMailWithAttachments(to, cc, bcc, replyTo, "some subject", "some body", docsId, "85f07f05-1641
        } 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**	Sends a mail message with attachments.

The value 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 it is sent.

Other parameters:

- from is optional and may be set to an empty value
- to is a list of destination email accounts.
- subject is the mail subject.
- cc, bcc, replyTo are lists of destination email accounts (optional)
- docsId is a list of valid document UUIDs already in OpenKM that will be sent as attachments in the 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", "8cd1e072-8595-4dd3-b12
            ws.mail.sendMailWithAttachments(from, to, cc, bcc, replyTo, "some subject", "some body", docsId, "85f07f05
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importEml

Description:

Method	Return values	Description
importEml(String uuid, String title, InputStream is)	**Mail**	Imports a mail in EML format.

The value 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 it 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", "some title", is);
            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**	Imports a mail in MSG format.

The value 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 it 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.msg");
            Mail mail = ws.mail.importMsg("85f07f05-1641-47e8-891f-0ea30643554e", "some title", is);
            System.out.println(mail);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(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.setTitle("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.toString());
        } 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.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

sendMail

Description:

Method	Return values	Description
sendMail(String to, String subject, String body)	**void**	Send a mail.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Test {

    private static final Logger log = LoggerFactory.getLogger(Test.class);

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

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

```
            // sendMail
            String to = "some@nomail.com";
            StringBuffer body = new StringBuffer();
            body.append("message test");
            ws.mail.sendMail(to, "sigma subject", body.toString());
        } catch (Exception e) {
            log.error(e.getMessage());
        }
    }
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(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.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(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.setDescription("b405a504-d8cb-4166-ac51-22f68acee8c5", "some description");
            Mail mail = ws.mail.getProperties("b405a504-d8cb-4166-ac51-22f68acee8c5");
            System.out.println("Description: " + mail.getDescription());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getContent

Description:

Method	Return values	Description
getContent(String mailId)	**InputStream**	Retrieves the 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.getContent("b405a504-d8cb-4166-ac51-22f68acee8c5");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
```

```
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

Method	Return values	Description
createMail(String uuid, String title, InputStream is, String type)	**WizardNode**	Creates a new document with a wizard.

The parameter **uuid** should be a valid folder or record **UUID**.

Available types:

- Mail.ORIGIN_MSG
- Mail.ORIGIN_EML

The WizardNode contains a list of pending actions that should be done to complete the document creation process. 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.createWizardl("4c195453-246b-4ce9-86ba-b84e68d1f284", "test title", is, Mail.OR
            System.out.print(wn);
            IOUtils.closeQuietly(is);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getThumbnail

Description:

Method	Return values	Description
getThumbnail(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/thumbnail.png");
            InputStream is = ws.mail.getMailThumbnail("1778f0b5-672c-48c1-b54e-494c18dd6df4", ThumbnailType.THI
            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, boolean orderAsc)	**SimpleNodeBaseResultSet**	Returns a paginated list of emails filtered by the values of the MailFilterQuery parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before beginning to return results.
- 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.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);
            MailFilterQuery filter = new MailFilterQuery();
            filter.setSubject("");
            filter.setAttachments(AttachmentEnum.ALL);
            String orderColumn = "subject";
            SimpleNodeBaseResultSet simpleNodeBaseResultSet = ws.mail.getMailsPaginated("/okm:root", 0, 20, filter,

            for (SimpleNodeBase simpleNodeBase : simpleNodeBaseResultSet.getResults()) {
                System.out.println(simpleNodeBase);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAccounts

Description:

Method	Return values	Description
getAccounts()	**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.getAccounts()) {
                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 on 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();
        }
    }
}
```

addAccount

Description:

Method	Return values	Description
addAccount(MailAccount mailAccount)	**void**	Adds an email account.

It is good practice to create an account and verify the mail configuration with testMailAccount(MailAccount mail).

When you do **not set the user**, the account will be **added by default to the logged-in 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.addAccount(mailAcount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateAccount

Description:

Method	Return values	Description
updateAccount(MailAccount mailAccount)	**void**	Updates 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.updateAccount(mailAcount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

testAccount

Description:

Method	Return values	Description
testAccount(MailAccount mailAccount)	**void**	Checks the 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.testAccount(mailAcount);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteAccount

Description:

Method	Return values	Description
deleteAccount(long mailAccountId)	**void**	Deletes 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 mailAccountId = 2;
            ws.mail.deleteAccount(mailAccountId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

importMessages

Description:

Method	Return values	Description
importMessages(long mailAccountId, List<Long> messageIds)	**void**	Imports messages from an email account.

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.mail.getMailMessages(mailAccountId, start);
            for (ExternalMail mail : msg.getServerMails()) {
                messageIds.add(mail.getUid());
            }

            ws.mail.importMailMessages(mailAccountId, messageIds);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFilter

Description:

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.createFilter(mailAccountId, filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateFilter

Description:

Method	Return values	Description
createFilter(long mailAccountId, MailFilter mailFilter)	**void**	Adds an email account filter.

Method	Return values	Description
updateFilter(MailFilter mailFilter)	**void**	Update an 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.updateFilter(filter);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteFilter

Description:

Method	Return values	Description
deleteFilter(long mailFilterId)	**void**	Delete an 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.deleteFilter(filterId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createRule

Description:

Method	Return values	Description
createRule(long filterId, MailFilterRule rule)	**void**	Create an 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 performed during the evaluation process.

Available options are:

- MailFilterRule.OPERATION_EQUALS
- MailFilterRule.OPERATION_NOT_EQUALS
- MailFilterRule.OPERATION_CONTAINS
- MailFilterRule.OPERATION_ENDS_WITH
- MailFilterRule.OPERATION_STARTS_WITH

- ***Value**: Sets the value that will be used for the evaluation process.*
- ***Active:** Sets whether the rule is enabled.*

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.createRule(filterId, rule);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateRule

Description:

Method	Return values	Description
updateMailRule(MailFilterRule rule)	**void**	Update an email account 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 performed during the evaluation process.

Available options are:

- MailFilterRule.OPERATION_EQUALS
- MailFilterRule.OPERATION_NOT_EQUALS
- MailFilterRule.OPERATION_CONTAINS
- MailFilterRule.OPERATION_ENDS_WITH
- MailFilterRule.OPERATION_STARTS_WITH

- ***Value**: Sets the value that will be used for the evaluation process.*
- ***Active:** Sets whether the rule is enabled.*

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

deleteRule

Description:

Method	Return values	Description
deleteRule(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.deleteRule(ruleId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFilterRules

Description:

Method	Return values	Description
getFilterRules(long filterId)	**List<MailFilterRule>**	Retrieve a list of rules for a 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.getFilterRules(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 an email to a list of users, roles, or external email addresses.

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.getProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
            InputStream is = ws.mail.getPdf(mail.getUuid());
            OutputStream fos = new FileOutputStream("/home/gnujavasergio/okm/" + mail.getSubject() + ".pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveAsPdf

Description:

Method	Return values	Description
saveAsPdf(String uuid, String newName, boolean propertyGroups, boolean security)	**Document**	Save the mail as a PDF in the OpenKM repository.

The value of the **uuid** parameter should be a Mail UUID.

When the newName parameter value is null, the document will keep 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.getProperties("7f6c5c0b-a66b-4782-9595-e14b402a18b0");
            Document docPdf = ws.mail.saveAsPdf(mail.getUuid(), mail.getSubject() + ".pdf", true, true);
            System.out.println("Document pdf: " + docPdf.getPath());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Node samples

Basics

Example of a UUID:

- Using UUID -> "**373bcdd0-c082-4e7b-addd-e10ef813946e**";

Suggested code sample

First, you must create the web service object:

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

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

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

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

At this point you can use all the Node methods from the "**node**" class as 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 for 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("3767deb4-21e7-4272-82be-fece5384fbab");
            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 the 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("3767deb4-21e7-4272-82be-fece53
            System.out.println(pre);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

promoteAsRecord

Description:

Method	Return values	Description
promoteAsRecord(String uuid)	**void**	Promote as 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.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 in an electronic record.

Returns 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("3767deb4-21e7-4272-82be-fece5384fbab"));
        } 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("3767deb4-21e7-4272-82be-fece5384fbab");
            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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

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, 10, "", NodesPaginationInfo.C
                    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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

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

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

- limit=10
- offset=0

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

- limit=10

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, 10, "", NodesPaginationInfo.C
                    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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

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(uuid, 0, 10, "", NodesPagi
                    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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

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

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

- limit=10

- offset=0

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

- limit=10
- offset=10

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(uuid, 0, 10, "", NodesPagi
                    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 the 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-91a7-24d2aa3f8837");
            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**	Deletes 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 a 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 uuid
            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>**	Returns 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**	Returns 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**	Generates a node download link.

When the parameter "preview" is set to true, the token will be generated for preview purposes.

The preview token expires by default after one minute.

The user must build the download URL with the returned token:

- Download: http://localhost:8080/openkm/download?DTK=token
- Preview: http://localhost:8080/openkm/download?PTK=token

Example:

```
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-abf76f9ec499", true));
        } 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)	**boolean**	Returns whether 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-b518-4fe7c353ee70")) {
             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", "Update comment");
        } 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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

```
ws.note.add("373bcdd0-c082-4e7b-addd-e10ef813946e", "the note text");
```

Methods

add

Description:

Method	Return values	Description
add(String uuid, String text)	**Note**	Adds a note to a node and returns a Note object.

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.add("373bcdd0-c082-4e7b-addd-e10ef813946e", "the note text");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

get

Description:

Method	Return values	Description
get(String noteId)	**Note**	Retrieves the note.

The noteId is a UUID.

The Node object has a variable named path; in that case, the path contains a UUID.

Example:

```
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.list("373bcdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                System.out.println(ws.get(notes.get(0).getPath()));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

delete

Description:

Method	Return values	Description
delete(String noteId)	**Note**	Deletes a note.

> The noteId is a UUID.
>
> The Node object has a variable named path; in that case, the path contains a UUID.

Example:

```
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.list("373bcdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.delete(notes.get(0).getPath());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

set

Description:

Method	Return values	Description
set(String noteId, String text)	**void**	Changes the note text.

The noteId is a UUID.

The Node object has a variable named path; in that case, the path contains a UUID.

Example:

```
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.list("373bcdd0-c082-4e7b-addd-e10ef813946e");
            if (notes.size() > 0) {
                ws.note.set(notes.get(0).getPath(), "text modified");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

list

Description:

Method	Return values	Description
list(String uuid)	**List<Note>**	Retrieves a list of all 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.list("373bcdd0-c082-4e7b-addd-e10ef813946e");
            for (Note note : notes) {
                System.out.println(note);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNoteHistories

Description:

Method	Return values	Description
getNoteHistories(String uuid)	**List<NoteHistory>**	Returns 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.getNoteHistories("3c68b3a1-c65c-4b1e-84b5-9ce2712ca573");
            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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

```
ws.notification.notify(uuids, users, roles, mails, message, false);
```

Methods

notify

Description:

Method	Return values	Description
notify(List<String> uuids, List<String> users, List<String> roles, List<String> mails, String message, boolean attachment)	**void**	Sends a mail notification.

The parameter uuids is a list of UUIDs for the nodes (document, folder, mail, or record).

The parameter users is a set of OpenKM users to be notified.

The parameter roles is a set of OpenKM roles to be notified.

The parameter mails is a set of email addresses (usually external addresses) to be notified.

The parameter message is the body content of the email.

When the attachment value is true, the node is attached to the email.

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

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

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

At this point you can use all the PDF methods from the "**pdf**" class as 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 the 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 1.

The **size** value is optional. When set, it is used to resize the image of the page; otherwise, the default value is used.

Example:

```
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", 1, null);
            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)	**List<Document>**	Returns a list of documents from the split pages.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of **baseName** is the base name used to create split documents. For example, if the baseName value is "test" and the pages value is "1,3", the files created will be "test-001.pdf" and "test-003.pdf".

The value of **pages** is the list of page numbers 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 com.openkm.sdk4j.bean.*;

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);
            List<Document> documents = ws.pdf.split("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09
            System.out.println(documents);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extract

Description:

Method	Return values	Description
extract(String uuid, String dstId, String name, List<Integer> pages)	**boolean**	Returns 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 list of page numbers 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-4e09-9c10-e7f27c731442", "e
                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)	**Document**	Returns a new document with the pages removed.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **pages** is the number of pages to be removed.

Example:

```
package com.openkm;

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

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

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);
            Document document = ws.pdf.extract("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c1
            System.out.println(document);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rotate

Description:

Method	Return values	Description
rotate(String uuid, String dstId, String name, String angle, List<Integer> pages)	**Document**	Returns a new document with the pages rotated.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the document.

The value of the **dstId** is the UUID of the destination (folder or record).

The value of **name** is the name of the new document with extracted pages.

The value of **angle** is the rotation applied to the chosen pages.

The value of **pages** is the number of pages to be rotated.

Example:

```
package com.openkm;

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

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

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);
            Document document = ws.pdf.rotate("e7d6860a-7e0b-4823-bd3d-75f88be1eedf", "0be057f1-efac-4e09-9c1(
            System.out.println("rotation done");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

insertPages

Description:

Method	Return values	Description
insertPages(String uuid, InsertPagesRequest request)	**Document**	Returns the new document created with the inserted pages.

The document must be a PDF or convertible to PDF.

The value of the **uuid** is the UUID of the target document where the pages will be inserted.

The value of **InsertPagesRequest** is an object that has List of **PageInsertOperation** named **insertOperations**, an String **dstId,** that contains the uuid of the destination folder, and a String **name** for the new generated document.

PageInsertOperation object has a String **srcId,** the UUID of the source file of the pages**,** a List of **Integer** named **pages** that stores the positions of the pages in the source PDF, and an int **insertFromPage,** the position where the pages have to be located in the target PDF

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.impl.OKMWebservices;
import com.openkm.sdk4j.bean.Document;
import com.openkm.ws.rest.request.InsertPagesRequest;
import com.openkm.ws.rest.request.PageInsertOperation;

import java.util.Arrays;

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);

            // Create the insert pages request
            InsertPagesRequest request = new InsertPagesRequest();
            request.name = "merged_document.pdf";  // Name for the new document
            request.dstId = "e7d6860a-7e0b-4823-bd3d-75f88be1eedf";

            // Create a page insertion operation
            PageInsertOperation operation = new PageInsertOperation();
            operation.srcId = "a573da75-ba1e-4f5b-98d2-fbb7da5f1cf1";  // Source document UUID
            operation.pages = Arrays.asList(1, 2, 3);  // Pages to insert from source
            operation.insertFromPage = 1;  // Insert at position 1 in target document

            request.insertOperations.add(operation);

            // Call insertPages with target document UUID and request
            Document result = ws.pdf.insertPages("6a54d98b-6454-4484-8172-8359a2b4f2f2", request);

            System.out.println("New document created: " + result.getUuid());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

optimize

Description:

Method	Return values	Description
optimize(String uuid)	**boolean**	Returns true when optimized successfully.

The UUID of the node must be a PDF 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);
            boolean success = ws.pdf.optimize("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
            System.out.println("optimize success: " + success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Plugin samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

```
ws.plugin.executePluginPost("com.openkm.plugin.rest.TestRestPlugin", parameters, String.class, null);
```

Methods

executePluginPost

Description:

Method	Return values	Description
executePluginPost(String className, Map<String, String> parameters, Class<?> clazz, InputStream is)	**Object**	Returns the Object value resulting from the execution of a class which implements RestPlugin.

- The "className" value must be the canonical class name of a class which implements the RestPlugin interface.
- The "parameters" map contains values that will be used by the class specified in className.
- The "clazz" value should be the Class of the returned object. It's used by REST for unmarshalling the result of the query.

- The "is" stream allows uploading a document.

This method executes a REST call via 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.rest.TestRestPlugin", parameters, S
            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executePostPluginReturnFile

Description:

Method	Return values	Description
executePluginPostReturnFile(String className, Map<String, String> parameters, InputStream is)	**InputStream**	Returns an InputStream resulting from the execution of a class which implements RestPlugin.

- The "className" value must be the canonical class name of a class which implements the RestPlugin interface.
- The "parameters" map contains values that will be used by the class specified in className.

- The "is" stream allows uploading a document.

This method executes a REST call via POST.

The RestPlugin must return a Response object. Take a look at the following sample implementation of the RestPlugin:

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestPlugin.class);

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
        }
```

```
            responseHeaders.setContentLength(doc.getActualVersion().getSize());
            InputStreamResource inputStreamResource = new InputStreamResource(isContent);
            log.debug("TestGetDocumentRestPlugin: [BINARY]");
            return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
        }
    }
```

Example:

```
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.rest.TestGetDocumentRestPlu
            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	Return
executePluginGet(String className, Map<String, String> parameters, Class<?> clazz)	**Object**

- The "className" value must be the canonical class name of a class which implements the RestPlugin interf

- The "parameters" map contains values that will be used by the class specified in className.
- The "clazz" value should be the Class of the returned object. It's used by REST for unmarshalling the result

This method executes a REST call via GET.

You can also execute it directly from the browser; use the following URL as a sample:

http://localhost:8080/OpenKM/services/rest/plugin/executeGetPlugin?className=com.openkm.plugin.

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?className=com.openkm.plugin.rest
            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.rest.TestRestPlugin", parameters, S
            System.out.println(value);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executePluginAtGetReturnFile

Description:

Method	Retu
executeGetPluginReturnFile(String className, Map<String, String> parameters)	**InputSt**

- The "className" value must be the canonical class name of a class which implements the RestPlugin interf
- The "parameters" map contains values that will be used by the class specified in className.

This method executes a REST call via GET.

The RestPlugin must return a Response object. Take a look at the following sample implementation of the RestPlug

```
package com.openkm.plugin.rest;

import net.xeoh.plugins.base.annotations.PluginImplementation;

import java.io.InputStream;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

import com.openkm.bean.Document;
import com.openkm.module.db.DbDocumentModule;
import com.openkm.plugin.BasePlugin;
import com.openkm.util.PathUtils;

/**
 * Sample rest plugin
 *
 * @author jllort
 */
@PluginImplementation
public class TestGetDocumentRestPlugin extends BasePlugin implements RestPlugin {

    private static Logger log = LoggerFactory.getLogger(TestGetDocumentRestPlugin.class);

    @Autowired
    private DbDocumentModule dbDocumentModule;

    @Autowired
    private PathUtils pathUtils;

    @Override
    public Object executePlugin(Map<String, String> parameters, InputStream is) throws Exception {
        log.debug("executePlugin({})", parameters);
        String docId = parameters.get("docId");
        boolean inline = parameters.containsKey("inline");
        Document doc = dbDocumentModule.getProperties(null, docId);
        String mimeType = doc.getMimeType();
        String fileName = pathUtils.getName(doc.getPath());
        InputStream isContent = dbDocumentModule.getContent(null, docId, false);

        HttpHeaders responseHeaders = new HttpHeaders();
        responseHeaders.add("Content-Type", mimeType);

        // inline true when you want to embedded the content into
        if (inline) {
            responseHeaders.add("Content-disposition", "inline; filename=\"" + fileName + "\"");
        } else {
            responseHeaders.add("Content-Disposition", "attachment; filename=\"" + fileName + "\"");
```

```
            }

            responseHeaders.setContentLength(doc.getActualVersion().getSize());
            InputStreamResource inputStreamResource = new InputStreamResource(isContent);
            log.debug("TestGetDocumentRestPlugin: [BINARY]");
            return new ResponseEntity<>(inputStreamResource, responseHeaders, HttpStatus.OK);
        }
    }
```

You can also execute it directly from the browser; use the following URL as a sample:

```
http://localhost:8080/OpenKM/services/rest/plugin/executeGetPluginReturnFile?className=com.open
```

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/executeGetPluginReturnFile?className=com.openkm.
            // http://localhost:8180/OpenKM/services/rest/plugin/executeGetPluginReturnFile?className=com.openkm.
            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.rest.TestGetDocumentRestPlu
            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 web service object:

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

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

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

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

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

```
ws.property.addCategory("eee9d70f-6af9-4783-82a7-b8ac94c234b6", "58c9b25f-d83e-4006-bd78-e26d7c6fb648");
```

Methods

addCategory

Description:

Method	Return values	Description
addCategory(String uuid, String catId)	**void**	Sets 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-d83e-4006-bd78-e26d7c6f
        } 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", "58c9b25f-d83e-4006-bd78-e26d7
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addKeyword

Description:

Method	Return values	Description
addKeyword(String uuid, String keyword)	**void**	Adds a keyword to a node.

The keyword should be a single word without spaces. Formats allowed:

- "test"
- "two_words" (the character "_" is used as a separator).

We also suggest that you add keywords 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 encrypted binary data in the repository.

The parameter nodeId should be a document node.

The parameter cipherName stores information about the encryption mechanism.

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

Example:

```
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 as normal binary data in the repository.

The parameter **uuid** should be a document node.

This method does not perform any kind of encryption; it simply marks in the database that a document is unencrypted.

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 in the repository.

The parameter **uuid** should be a document node.

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

Example:

```
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 whether the document is signed.

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("6330d2a0-529f-4c14-baa1-ce6a9200
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

PropertyGroup samples

Basics

In older OpenKM versions we called "**Metadata Groups**" "**Property Groups**".

Although we understand this name does not help much to identify these methods as metadata-related, for historical reasons we continue to maintain the nomenclature.

For more information about Metadata.

The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields. Metadata field values of type date are 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 date
```

Suggested code sample

First, you must create the web service object:

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

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

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

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

At this point you can use all the Property Group methods from the "**propertyGroup**" class as shown below:

```
ws.propertyGroup.addGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", propertiesMap);
```

Methods

addGroup

Description:

Method	Return values	Description
addGroup(String uuid, String grpName, Map<String, String> propertiesMap)	**void**	Adds an empty metadata group to a node.

The grpName should be a valid Metadata group name.

It is not mandatory to set all field values in the propertiesMap parameter; it is enough to include the fields whose values you wish to change.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="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 to 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 the character "**,**" is used as a 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.addGroup("4a3b1c1b-c880-45a3-a6ff-2c8b7c5adfa5", "okg:technology", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeGroup

Description:

Method	Return values	Description
removeGroup(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.removeGroup("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroups

Description:

Method	Return values	Description
getGroups(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.getGroups("8cd1e072-8595-4dd3-b121-41d622c43f08")) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllGroups

Description:

Method	Return values	Description
getAllGroups()	**List<PropertyGroup>**	Retrieves a list of all metadata groups defined in 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.getAllGroups()) {
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getAllGroups

Description:

Method	Return values	Description
getAllGroups(String uuid)	**List<PropertyGroup>**	Retrieves a list of all metadata groups defined in the application.

This method allows, through the "Get all metadata groups" automation task event, filtering the list of groups allowed in a specific node.

Example:

```
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.getAllGroups("8cd1e072-8595-4dd3-b121-41d622c43f08")) {
                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 element definitions.

The grpName should be a valid Metadata group name.

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

Example:

```
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:consulting")) {
                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 element definitions.

The grpName should be a valid Metadata group name.

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

Example:

```
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.getPropertyGroupFormDefinition("okg:consulting")) {
                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 element definitions.

The grpName should be a valid Metadata group name.

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

Example:

```
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.getPropertyGroupFormDefinition("okg:consulting", "8cd1e07
                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("8cd1e072-8595-4dd3-b121-41d622
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setProperties

Description:

Method	Return values	Description
setProperties(String uuid, String grpName, Map<String, String> properties)	**void**	Changes the metadata group values of a node.
The grpName should be a valid Metadata group name.		

Before changing metadata, you **should have the group added to the node** (see addGroup method); otherwise an error will be raised.

It is not mandatory to set all field values in the properties parameter; it is enough to include the fields whose values you wish to change.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="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 to 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 the character "**,**" is used as a 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.setProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:consulting", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasGroup

Description:

Method	Return values	Description
hasGroup(String uuid, String grpName)	**Boolean**	Returns a boolean that indicates whether the node has 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.hasGroup("8cd1e072-8595-4dd3-b121-41d6
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getRegisteredDefinition

Description:

Method	Return values	Description
getRegisteredDefinition()	**String**	Returns the XML Metadata groups definition.

The method can only be executed by a super user (ROLE_ADMIN member).

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

registerDefinition

Description:

Method	Return values	Description

registerDefinition(InputStream is, String name)	**void**	Sets the XML Metadata groups definition in the repository.

The method can only be executed by a super user (ROLE_ADMIN member).

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.registerDefinition(is, "test");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestions

Description:

Method	Return values	Description
getSuggestions(String uuid, String grpName, String propName)	**List<String>**	Retrieves a list of suggested metadata field values.

The propName parameter should be a Metadata Select field type.

More information at Creating your own Suggestion Analyzer and Metadata Select field.

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 2.0//EN"
                                 "http://www.openkm.com/dtd/property-groups-2.0.dtd">
<property-groups>
  <property-group label="Technology" name="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.getSuggestions("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:techr
                System.out.println(value);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	Return values	Description
getProperties(String uuid, String	**Map<String,**	Retrieves a map - (key, value) pairs - with Metadata group

grpName)	**String>**	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.getProperties("8cd1e072-8595-4dd3-b121-41d622c43f08", "okg:technology")

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroupsByVersion

Description:

Method	Return values	Description
getGroupsByVersion(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.getGroupsByVersion("118cb06b-9df7-4d59-bed9-ba986a4f8
                System.out.println(pGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPropertiesByVersion

Description:

Method	Return values	Description
getPropertiesByVersion(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.getPropertiesByVersion("118cb06b-9df7-4d59-bed9-ba986a4f8e0b", "okg:tec

            for (String key : properties.keySet()) {
                System.out.println(key + " > " + properties.get(key));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroupByVersionForm

Description:

Method	Return values	Description
getGroupByVersionForm(String uuid, String grpName, String versionName)	**List<FormElement>**	Retrieves a list of all metadata group element definitions 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 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.getGroupByVersionForm("118cb06b-9df7-4d59-bed9-ba986
                System.out.println(fElement);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroup

Description:

Method	Return values	Description
getGroup(String grpName)	**PropertyGroup**	Retrieves 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.getGroup("okg:consulting"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getSuggestBoxKeyValue

Description:

Method	Return values	Description
getSuggestBoxKeyValue(String grpName, String propertyName, String key)	**String**	Returns the suggestBox value for key.

The propertyName parameter should be a Metadata Suggestbox field type.

More information at Metadata Suggestbox field

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.7.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
     <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
       filterQuery="select ct_id, ct_name from country where ct_name like '%{0}%' order by ct_name"
       valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Example:

```
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	Description
getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)	**Map<String, String>**	Retrieves a map - (key, value) pairs - wit suggestBox values.

The propertyName parameter should be a Metadata Suggestbox field type.

More information at Metadata Suggestbox field

The sample below is based on this Metadata group definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE property-groups PUBLIC "-//OpenKM//DTD Property Groups 3.7//EN"
                                 "http://www.openkm.com/dtd/property-groups-3.7.dtd">
<property-groups>
  <property-group label="Consulting" name="okg:consulting">
     <suggestbox label="country" name="okp:consulting.suggestbox"
     width="200px" dialogTitle="Choose country" filterMinLen="3"
       filterQuery="select ct_id, ct_name from country where ct_name like '%{0}%' order by ct_name"
       valueQuery="select ct_id, ct_name from country where ct_id='{0}'" />
  </property-group>
</property-groups>
```

Example:

```
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.getSuggestBoxKeyValuesFiltered(Config.GROUP_CONSUL
            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**	Retu

- The "className" value must be the canonical name of a class that implements the FieldValidator interface.

Example of a form validator implementation

More information is available at Creating your own Form Validator plugin.

In this example, two identical comments will not be allowed in the metadata field named okp:technology.comment.

```
package com.openkm.plugin.form.validator;

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

import com.openkm.module.db.stuff.DbSessionManager;
import com.openkm.plugin.form.FieldValidator;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import com.openkm.api.OKMRelation;
import com.openkm.bean.Relation;
import com.openkm.db.service.LegacySrv;
import com.openkm.plugin.BasePlugin;

import net.xeoh.plugins.base.annotations.PluginImplementation;

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements FieldValidator {

    @Autowired
    private LegacySrv legacySrv;

    @Autowired
    private OKMRelation okmRelation;

    @Override
    public String getName() {
        return "Duplicated document number";
    }

    @Override
    public String validate(String value, List<String> uuids) {
        String validate = "";

        String token = DbSessionManager.getInstance().getSystemToken();
        String sql = "SELECT RGT_UUID FROM OKM_PGRP_CUR_TECHNOLOGY WHERE RGT_PR
        try {
            List<List<String>> result = legacySrv.executeSQL(sql);
            if (result.size() > 0) {
                if (uuids.isEmpty()) {
                    validate = "Duplicated document number";
                } else {
                    List<String> allowedUuids = new ArrayList<>();
                    for (String uuid : uuids) {
                        allowedUuids.add(uuid);
                        List<Relation> relations = okmRelation.getRelations(token, uuid);
                        for (Relation relation : relations) {
                            allowedUuids.add(relation.getNode());
                        }
                    }

                    for (List<String> resultList : result) {
                        if (!allowedUuids.contains(resultList.get(0))) {
                            validate = "Duplicated document number";
                        }
                    }
                }
            }
        } catch (Exception e) {
            validate = e.getMessage();
        }

        return validate;
    }
}
```

Example

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

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

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

At this point you can use all the Record methods in the "**record**" class as shown below:

```
ws.record.create("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new title", 0)
```

Methods

create

Description:

Method	Return values	Description
create(String uuid, String name, String title, long nodeClass)	**Record**	Creates a new record and returns a Record object.
The parameter **uuid** should be a 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.create("ada67d44-b081-4b23-bdc1-74181cafbc5d", "PKI-100200", "new title", 0);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProperties

Description:

Method	Return values	Description
getProperties(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.getProperties("fbe2933e-b141-4557-ab7a-736820ecdb2e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(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.delete("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

purge

Description:

Method	Return values	Description
purge(String uuid)	**void**	The record is definitively removed from the repository.

Usually you will purge records to /okm:trash/userId - the personal trash user location - but it is possible to directly purge any record from the whole repository.

When a record is purged it can only be restored from a previous repository backup. The purge action removes the record permanently 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.purge("fbe2933e-b141-4557-ab7a-736820ecdb2e");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

rename

Description:

Method	Return values	Description
rename(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.rename("5489fc37-3eb7-43de-998c-319725ae0ca0", "new_name");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

move

Description:

Method	Return values	Description

move(String uuid, String dstId)	**void**	Moves a record to a folder or another record.
The value 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.move("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d6950c63");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

copy

Description:

Method	Return values	Description
copy(String uuid, String dstId, String newName)	**void**	Copies a record to a folder or another record.

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

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

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

Example:

```
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.copy("5489fc37-3eb7-43de-998c-319725ae0ca0", "8599eab7-ae61-4628-8010-1103d6950c63", "
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

isValid

Description:

Method	Return values	Description
isValid(String uuid)	**Boolean**	Returns a boolean that indicates whether the node is 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);
            System.out.println("Is a record:" + ws.record.isValid("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getChildren

Description:

Method	Return values	Description
getChildren(String uuid)	**List<Record>**	Returns a list of all records whose parent is fldId

The parameter uuid can be a folder or a record node.

Example:

```
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.getChildren("8599eab7-ae61-4628-8010-1103d6950c63")) {
                System.out.println(rec);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setTitle

Description:

Method	Return values	Description
setTitle(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.setTitle("5489fc37-3eb7-43de-998c-319725ae0ca0", "some title");
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

getPath

Description:

Method	Return values	Description
getPath(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.getPath("5489fc37-3eb7-43de-998c-319725ae0ca0"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setNodeClass

Description:

Method	Return values	Description
setNodeClass(String uuid, long ncId)	**void**	Sets 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.setNodeClass("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", ncId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDispositionStage

Description:

Method	Return values	Description
setDispositionStage(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.setDispositionStage("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", stage);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setDescription

Description:

Method	Return values	Description
setDescription(String uuid, String description)	**void**	Sets 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.setDescription("7ce1b4a8-4ade-4dce-8d7d-4e99a6cd368b", "some description");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

extendedCopy

Description:

Method	Return values	Description
extendedCopy(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 a folder or record.

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

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

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

Additional:

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

- When the security parameter is true, the original value of the security will be copied.

Example:

```
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.extendedCopy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-46
                    "new name record", true, true, true, true, true);
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createWizard

Description:

Method	Return values	Description
createWizard(String uuid, String name, String title, long nodeClass)	**WizardNode**	Creates a new record using a wizard.

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

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

- Add keyword
- Add Categories
- Add Metadata

Example:

```
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.createWizard("1f323e88-64ee-4f57-91e2-9276f8c603f9", "PKI-100200", "new ti
            System.out.print(wn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createFromTemplate

Description:

Method	Return values	Description
createFromTemplate(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 in the properties parameter.

Additional:

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

Example:

```
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.createFromTemplate("9a114b17-7e51-41e7-9d3a-dc7b77e37656", "/okm:root/sd
            System.out.println(record);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createMissingRecords

Description:

Method	Return values	Description
createMissingRecords(String recPath)	**String**	Creates 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/missingrec3");
        } 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 you should log in using the method "**login**." You can access the "**login**" method from the web service object "**ws**" as shown below:

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

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

At this point, you can use all the Relation methods from the " **relation**" class as shown below:

```
ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL);
```

Methods

getRelationTypes

Description:

Method	Return values	Description
getRelationTypes(String type)	**List<RelationType>**	Retrieves a list of all relations defined for a type.

Available types:

- RelationType.BIDIRECTIONAL
- RelationType.PARENT_CHILD
- RelationType.MANY_TO_MANY

More information on Relation types.

Example:

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

add

Description:

Method	Return values	Description
add(String nodeAId, String nodeBId, long relTypeId)	**void**	Sets a relation between two nodes.
The parameters **nodeAId** and **nodeBId** should be any valid document, folder, mail, or record **UUID**.		

Example:

```
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.add("46762f90-82c6-4886-8d21-ad3017dd78a7", "8cd1e072-8595-4dd3-b121-41d622c43f0
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(long relationId)	**void**	Deletes a relationship.

A relationship can only be deleted when no node uses it. 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-ad3017dd78a7")) {
                // looking for a relation named invoice
                if (relation.getRelationTitle().equals("invoice")) {
                    ws.relation.delete(relation.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

getRelations

Description:

Method	Return values	Description
getRelations(String uuid)	**List<Relation>**	Retrieves a list of all relations for 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("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                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 relation groups for 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("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                System.out.println(rg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addGroup

Description:

Method	Return values	Description
addGroup(String uuid, String groupName, long type)	**void**	Adds a relation group to a node.

A relation group only makes sense for the relation type 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_TO_MANY)) {
                if (type.getTitle().equals("staple")) {
```

```
                ws.addGroup("46762f90-82c6-4886-8d21-ad3017dd78a7", "staple group", type.getId());
            }
        }
    } 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.

A relation group is only applicable to the relation 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("46762f90-82c6-4886-8d21-ad3017dd78a7")) {
                if (rg.getName().equals("staple group")) {
                    ws.relation.addNodeToGroup("8f101a85-88e7-4abe-8175-c5fea3e17d8b", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteGroup

Description:

Method	Return values	Description
deleteGroup(long groupId)	**void**	Removes 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.getGroups("8f101a85-88e7-4abe-8175-c5fea3e17d8b")) {
                if (rg.getName().equals("staple group")) {
                    ws.relation.deleteRelationGroup(rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getGroup

Description:

Method	Return values	Description
getGroup(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.getGroup(groupId));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setGroupName

Description:

Method	Return values	Description
setGroupName(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.setGroupName(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 parameters "limit" and "offset" allow you to retrieve just a portion of the results of a query.

- The parameter "limit" limits the number of results returned.
- The parameter "offset" specifies how many results to skip before starting 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 to 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(relationTypeId, filter, 0, 10);
            System.out.println("Total: " + resultSet.getTotal());
            for (RelationGroup relationGroup : resultSet.getResults()) {
                System.out.println(relationGroup);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteItemFromGroup

Description:

Method	Return values	Description
deleteItemFromGroup(String uuid, long groupId)	**void**	Removes a node from a relation 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("930baee0-8712-41b0-85c0-81d21e55742f")) {
                if (rg.getName().equals("staple")) {
                    ws.relation.deleteItemFromGroup("930baee0-8712-41b0-85c0-81d21e55742f", rg.getId());
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Report samples

Basics

The table below shows how variables should be passed based on field type:

Field type	Type	Description
Date	**String**	Use the pattern yyyy-MM-dd (year-month-day). 2018-10-04
Select multiple	**String**	Use ',' to separate each value. "value1,value2,value3"

Suggested code sample

First, you must create the webservice object:

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

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

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

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

At this point, you can use all the Report methods from the "**report**" class as shown below:

```
ws.report.getReports(true)
```

Methods

getReports

Description:

Method	Return values	Description
getReports(boolean active)	**List<Report>**	Returns a list of reports.

Example:

```
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 report.

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**	Returns 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();
        }
    }
}
```

execute

Description:

Method	Return values	Description
execute(long rpId, Map<String, String> params, String format, String uuid)	**InputStream**	Returns a document resulting from executing a report.

Available formats:

- Report.FORMAT_CSV

- Report.FORMAT_DOCX
- Report.FORMAT_HTML
- Report.FORMAT_ODT
- Report.FORMAT_PDF
- Report.FORMAT_RTF
- Report.FORMAT_TEXT

The parameter **uuid** is the UUID of a node (this parameter is optional; it is usually meaningful for reports executed from the user interface where the results depend on the selected node).

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.execute(rpId, params, format, dstId, uuid);
            OutputStream fos = new FileOutputStream("/home/openkm/report/document Create.pdf");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeSql

Description:

Method	Return values	Description
executeSql(long rpId)	**InputStream**	Returns a document resulting from executing a report SQL query.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
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);

            // executeReportSql
            long rpdId = 1;
            InputStream is = ws.report.executeSql(rpdId);
            OutputStream fos = new FileOutputStream("/home/openkm/report/activity.csv");
            IOUtils.copy(is, fos);
            IOUtils.closeQuietly(is);
            IOUtils.closeQuietly(fos);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

save

Description:

Method	Return values	Description
save(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 **dstId** 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; it is usually meaningful for reports executed from the user interface where the results depend on the selected node).

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.save(rpId, params, format, dstId, docName, uuid);
            System.out.println(documenReport);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveSql

Description:

Method	Return values	Description
saveSql(long rpId, Map<String, String> params, String dstId, docName)	Document	Execute the report SQL and save the resulting CSV document.

> The values of the **dstId** parameter should be a folder or record UUID.
>
> The parameter **docName** is the file name of the report.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.Document;
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);

            // saveSql
            long rpId = 1;
            String dstId = "8f64d889-1600-4cbc-8245-de6fb62214eb";
            String docName = "activity.csv";
            Document documenReport = ws.report.saveSql(rpId, new HashMap<>(), dstId, docName);
            System.out.println(documenReport);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

getSqlReports

Description:

Method	Return values	Description
getSqlReports(boolean active)	**List<Report>**	Returns a list of SQL reports.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
import com.openkm.sdk4j.bean.SqlReport;
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);

            // getSqlReports
            List<SqlReport> sqlReports = ws.report.getSqlReports(true);
            for (SqlReport sqlReport : sqlReports) {
                System.out.println(sqlReport);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Repository samples

Basics

Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

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

Methods

getRootFolder

Description:

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

The returned folder will be the user's 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 Folder object for the 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 Folder object for the 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 Folder object for the node "/okm:personal/{userId}".

The returned folder will be the user's 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 Folder object for the 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 Folder object for the node "/okm:mail/{userId}".

The returned folder will be the user's 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 Folder object for the 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();
        }
    }
}
```

getCategoriesFolder

Description:

Method	Return values	Description
getCategoriesFolder()	**Folder**	Returns the Folder object for the 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**	Permanently removes from the repository all nodes in "/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 can only be restored from a previous repository backup. The purge action permanently removes the node 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 a new OpenKM release is available.

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

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

Example:

```
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's 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 boolean indicating whether a node exists.
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-c082-4e7b-addd-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodePath

Description:

Method	Return values	Description
getNodePath(String uuid)	String	Converts a node UUID to a 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-e10ef813946e"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNodeUuid

Description:

Method	Return values	Description
getNodeUuid(String nodePath)	**String**	Converts a node path to a 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 the 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 one node to another.

The value of the dstId parameter should be a node UUID.

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

- When the propertyGroups parameter is true, the original values of the metadata groups will be copied.
- When the notes parameter is true, the original values of the notes will be copied.

Example:

```
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", "ac9fe744-8e45-4bf4-a293-85daf
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

executeScript

Description:

Method	Return values	Description
executeScript(InputStream is)	**ScriptExecutionResult**	Executes a 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 superuser (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 a script.

This action can only be done by a superuser (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) ContextWrapper.getContext();"
                    + " OKMFolder okmFolder = cc.getBean(OKMFolder.class);"
                    + " for (Folder fld : okmFolder.getChildren(null,\"/okm:root\")) {"
                    + " print(fld.getPath());"
                    + "}";
            ScriptExecutionResult result = ws.repository.executeScript(script);
            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 statements.

The test.sql script used in the sample below:

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

The SQL script can only contain a single SQL statement.

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

Example:

```
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 statements.

The SQL script can only contain a single SQL statement.

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

Example:

```
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, NBS_CONTEXT, NBS_N
            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 statements.

The test.sql script used in the sample below:

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

The HQL script can only contain a single HQL statement.

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

Example:

```
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: " + row.getColumns().get(1));
            }

            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, author from NodeBase where nam
            for (HqlQueryResultColumns row : result.getResults()) {
                System.out.println("uuid: " + row.getColumns().get(0) + ", name: " + row.getColumns().get(1));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTranslations

Description:

Method	Return values	
getTranslations(String lang, String module)	**Map<String, String>**	Retrieves the

The OpenKM translations tables can be used to retrieve the actual OpenKM translations or create your own translat

By default, module values are:

- frontend (used by the default OpenKM UI).
- extension (used by the OpenKM extension UI).
- mobile (used by the OpenKM mobile UI).

Example to add a new translation module:

SQL 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", "frontend");
            for (String key : translations.keySet()) {
                System.out.println("key:'" + key + "', with translation:'" + translations.get(key) + "'");
            }
        } 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 has the configuration parameter named "**webservices.visible.properties**", access to configuration parameters will be restricted for non-administrator users. That means any non-administrator user who tries to access configuration parameters through the webservices that are not listed in "**webservices.visible.properties**" will receive 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>**	Returns the list of changes in a path and its subfolders.

- The method is used by the desktop synchronization application for retrieving 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", now)) {
                System.out.println(cl);
            }
```

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

getServerTime()

Description:

Method	Return values	Description
getServerTime()	**String**	Returns the current server time.

The server time returned is in ISO8601 format.

- The method is used by the desktop synchronization application for retrieving 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>**	Returns 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**	Returns 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	Returns the 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();
        }
    }
}
```

Search samples

Basics

Almost all methods use QueryParams. Here are some tips about how to use it.

Variables	Type	Allow wildcards	Restrictions
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; do not use a 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, the "/okm:root" node is used by default. Value should be a valid UUID; do not use a path value.
folderRecursive	**Boolean**	No.	It only makes sense to set this variable to true when the folder variable i
lastModifiedFrom	**Calendar**	No.	
lastModifiedTo	**Calendar**	No.	
mailSubject	**String**	Yes.	Only applies to mail nodes.

mailFrom	**String**	Yes.	Only applies to mail nodes.
mailTo		Yes.	Only applies to mail nodes.
notes		Yes.	

properties	**Map<String, String>**	Yes for almost all.	On metadata field values like "date", wildcards cannot be applied.
			
			The map of the properties is composed of pairs:
			
			('metadata_field_name','metadata_field_value')
			
			For example:
			
			`Map<String, String> properties = new HashMap();`
			`properties.put("okp:consulting.name", "name value");`
			
			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 before`
			`Map<String,String> properties = new HashMap<>();`
			`properties.put("okp:consulting.date", ISO8601.formatBasic(fron`
			
			When filtering by a range of dates you must set both values will be ignored by OpenKM.
			
			To filter by a metadata field of type multiple like this:
			
			`<select label="Multiple" name="okp:consulting.multiple" type="`
			`<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;two");
```

Where "**one**" and "**two**" are valid values and the character "**;**" is used as

The search operation is performed using only AND logic.

Wildcard examples:

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

Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

```
ws.search.find(qParams, null)
```

Methods

find

Description:

Method	Return values	Description
find(QueryParams queryParams, String propertiesPlugin)	**List<QueryResult>**	Returns a list of results filtered by the values of the queryParams parameter.

The parameter "propertiesPlugin" must be a canonical class name of the class which implements the NodeProperties interface.

Retrieving entire objects (Document, Folder, Record, Mail) from REST can take a lot of time - marshalling and unmarshalling - while you are only interested in using a few object variables. If this is your case, you can use NodeProperties classes to retrieve the object variables 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 - while you are only interested in using a few object variables. If this is your case, you can use NodeProperties classes to retrieve the object variables 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_MODIFIED, true, null)) {
                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 60â€“70% faster than the other search methods (these metrics should be taken as indicative values because they depend on hardware and the specific scenario where the application is running). The other search methods return objects of type Node, which come with full node data. The SimpleNodeBase object provides less data but in most cases should be enough. Especially when the limit is a higher value, you will appreciate the difference in performance.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
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, 20, 10);
            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(params, SearchSortField.AU
            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
findSimpleNodeBasePaginated(QueryParams queryParams, String sortField, boolean sortReverse, int offset, int limit, String pluginName)	**SimpleNodeBaseResultSet**	Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter.

Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods (these metrics should be taken as 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(params, SearchSortField.AU
            System.out.println("Total: " + result.getTotal());
            for (SimpleNodeBase nodeBase : result.getResults()) {
                if (nodeBase != null) {
                    System.out.println(nodeBase.toString());
                }
            }
        } 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-bd78-e26d7c6fb648")) {
                System.out.println(doc);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

saveSearch

Description:

Method	Return values	Description
saveSearch(QueryParams params)	**Long**	Saves a search parameters.
The queryName variable of the params parameter must 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 previously saved search parameters.

Only a saved search created by the same user who is executing the method can be updated.

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();
        }
    }
}
```

getAllSearches

Description:

Method	Return values	Description
getAllSearches()	**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.getAllSearches()) {
                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.plugin.search.TestNode
            System.out.println(nodeSearchConfig);
        } 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:t*.pdf", null);
            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:t*.pdf", SearchSortField.NAMI
            for (QueryResult qr : results) {
                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.findSimpleNodeBaseByQueryPaginated("text:grial AND name
            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.findSimpleNodeBaseByQueryPaginated("text:grial AND name
            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.NAME, true, null, groups)) {
                System.out.println(qr);
            }

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

findWithMetadataPaginated

Description:

Method	Return values	Description
findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List<String> groups)	**ResultSet**	Returns a list of paginated results with metadata values filtered by the queryParams parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning 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 - the marshalling and unmarshalling processes - while you are only interested in using a few object variables. If it's your case you can use NodeProperties classes for retrieving the object variables that 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, groups);
            System.out.println("Total results: " + rs.getTotal());

            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findWithMetadataPaginated

Description:

Method	Return values	Description
findWithMetadataPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin, List<String> groups)	**ResultSet**	Returns a list of paginated results with metadata values filtered by the queryParams parameter.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before returning 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 - the marshalling and unmarshalling processes - while you are only interested in using a few object variables. If it's your case you can use NodeProperties classes for retrieving the object variables that 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, SearchSortField.AUTHOR, true, 0, 10, null, g
            System.out.println("Total results: " + rs.getTotal());
            for (QueryResult qr : rs.getResults()) {
                System.out.println(qr);
            }
        } 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 639-1 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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

```
ws.shard.getShards()
```

Methods

getShards

Description:

Method	Return values	Description
getShards()	**List<Shard>**	Returns a list of all shards.

Example:

```
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 shard ID 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();
        }
    }
}
```

Stamp samples

Basics

Suggested code sample

First, you must create the web service object:

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

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

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

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

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

```
ws.stamp.getAllStamps();
```

Methods

getAllStamps

Description:

Method	Return values	Description
getAllStamps()	**List<StampItem>**	Returns a list of stamp items.

Example:

```
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**	Returns 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-6ceb-4f94-bf20-9ba2405c5c
            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**	Returns 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, "928de61e-6ceb-4f94-bf20-9ba2
            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**	Returns 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, "6330d2a0-529f-4c14-b
            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**	Returns the calculated stamp coordinates.

In the 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 on the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

The parameter **stampAlign**.

Available values:

- text

- image

The parameter **stampAlign** is the text alignment.

Available values:

- left
- center
- right

Example:

```
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(imgToStampWidth, imgToSta
                    floatingDivWidth, floatingDivHeight, exprX, exprY, stampType, stampAlign);
            System.out.println("X = " + stampCoordinates.getX() + ", Y = " + stampCoordinates.getY());

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

calculateStampExpressions

Description:

Method	Return values	Description

calculateStampExpressions(long imgToStampWidth, long imgToStampHeight, long posX, long posY)	**StampExpressions**	Returns the calculated stamp expressions for X and Y.

Example:

```
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(imgToStampWidth, imgToSta
            System.out.println("X = " + stampExpressions.getExprX() + ", Y = " + stampExpressions.getExprY());

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

stampText

Description:

Method	Return values	Description
stampText(String uuid, long id)	**void**	Stamps 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**	Stamps an 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**	Stamps a 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", stampBarcodeId);
        } 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 a document with an image that is in OpenKM

The **id** is the ID of an image stamp created by the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

The **personalStampUuid** must be a valid document **UUID**. The document must be an image (jpg or png).

In the 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 on 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, exprX, exprY, "", "ac9cdfb4
        } 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 created by the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

In Expr. X and Expr. Y input fields you can put more than a simple number. Currently, the following macros are defined:

- IMAGE_WIDTH
- IMAGE_HEIGHT
- PAGE_WIDTH
- PAGE_HEIGHT
- PAGE_CENTER
- PAGE_MIDDLE

So to center a stamp in the page you can use:

Expr. X	PAGE_CENTER
Expr. Y	PAGE_MIDDLE

Example:

```
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", stampTextId, text, exprX, exprY,
        } 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 created by the administrator.

The **exprX** is the expression to set the position in X coordinates.

The **exprY** is the expression to set the position in Y coordinates.

In the 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 on 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, exprX, exprY, "", 0, is);
        } 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**	Returns the calculated height and width for the image to stamp.

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("babe24df-c443-49a5-9453-3
            System.out.println(stampImageSize);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getPersonalStamps

Description:

Method	Return values	Description
getPersonalStamps()	**List<Document>**	Returns a list of personal seals.

Returns a list of documents of type image (jpg or png) from /okm:templates/userId/stamps

Example:

```
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**	Returns the processed personal seal.

When stamping a document with an image, the image can also have a text layer.

This method processes the image and returns an image with the text layer.

The **uuid** must be a **UUID** returned by the method getPersonalStamps.

Example:

```
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("928de61e-6ceb-4f94-bf2(
            System.out.println(stampPersonalImage);
        } 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 the application.	Not applicable.
Owner	**String**	It contains the OpenKM user ID. It is automatically set by the application.	Not applicable.
Subject	**String**	The topic of the task.	Yes.
Description	**String**	The description of the task.	No.
Start	**Calendar**	When the task might start.	Yes.
End	**Calendar**	When the task might end.	No.
Status	**TaskStatus**	The task status. You administer the list of status values.	Yes.
Project	**TaskProject**	The project related with the task. You administer the list of project values.	Yes.

Type	**TaskType**	The task type. You administer the list of task type values.	Yes.
Progress	**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 over time it is a member of a group. The group is identified by a unique repeat group ID.	No.
ReminderStartValue	**Integer**	How many days before the task starts, the system might send an email notification to the user.	No.
ReminderEndValue	**Integer**	How many days before the task ends, the system might send an email notification to the user.	No.
ReminderStartUnit	**String**	Reminder start units. Allowed values: • "m" for minutes. • "h" for hours. • "d" for days.	Mandatory when **ReminderStartValue** is greater than 0.
ReminderEndUnit	**String**	Reminder end units. Allowed values: • "m" for minutes.	Mandatory when **ReminderEndValue** is greater than 0.

		• "h" for hours. • "d" for days.	
Users	**Set<String>**	Collection of users assigned to the task.	No. But at least one user or role should be assigned.
Roles	**Set<String>**	Collection of roles assigned to the task.	No. But at least one user or role should be assigned.
Documents	**Set<String>**	List of UUIDs of the related documents.	No.
Folders	**Set<String>**	List of UUIDs of the related folders.	No.
Mails	**Set<String>**	List of UUIDs of the related mails.	No.
Records	**Set<String>**	List of UUIDs of the related records.	No.

On all methods, you'll see a parameter named "**token**". When accessing the application via SOAP, the login process returns a token, which is used to identify the user on all the exposed methods. By default, from the application execution context you must use the "**null**" value, which indicates the application must use the "**user session**".

In special cases, you might be "**promoted as Administrator**" using the "**administrator token**".

```
String systemToken = DbSessionManager.getInstance().getSystemToken();
```

Suggested code sample

First, you must create the webservice object:

```
OKMWebservices ws = OKMWebservicesFactory.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 in with the webservices, the session is kept in the webservice object. Then you can use the other API methods.

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

```
ws.task.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, subject);
```

Methods

getAssigned

Description:

Method	Return values	Description
getAssigned(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 parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The 'from' date to filter (this parameter is optional)
- The 'to' date to filter (this parameter is optional)
- The subject parameter: the topic of the task.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before starting 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 parameter name used in HQL query
            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.getAssigned(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, ""
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getActive

Description:

Method	Return values	Description
getActive(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 parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The 'from' date to filter (this parameter is optional)
- The 'to' date to filter (this parameter is optional)
- The subject parameter: the topic of the task.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before starting 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 parameter name used in HQL query
            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.getActive(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "");
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinished

Description:

Method	Return values	Description
getFinished(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 parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.
- The orderAsc parameter orders ascending or descending.
- The 'from' date to filter (this parameter is optional)
- The 'to' date to filter (this parameter is optional)
- The subject parameter: the topic of the task.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before starting 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 parameter name used in HQL query
            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.getFinished(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "")
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotified

Description:

Method	Return values	Description
getNotified(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 parameter of the TaskManagerTask class. Commonly used parameters are "subject", "start", "end", "progress", "owner". More information at javadoc documentation.

- The orderAsc parameter orders ascending or descending.
- The 'from' date to filter (this parameter is optional)
- The 'to' date to filter (this parameter is optional)
- The subject parameter: the topic of the task.

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" specifies how many results to skip before starting 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 parameter name used in HQL query
            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.getNotified(projectId, typeId, statusId, orderColumn, true, 0, 10, from, to, "");
            for (Task task : listAssigned.getTasks()) {
                System.out.println(task);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStatus

Description:

Method	Return values	Description
getStatus()	**List<TaskStatus>**	Retrieve a list of all the task statuses.

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

getProjects

Description:

Method	Return values	Description

getProjects(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.getTProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTypes

Description:

Method	Return values	Description
getTypes(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.getTypes(true)) {
```

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

getAssignedCount

Description:

Method	Return values	Description
getAssignedCount(long statusId, long projectId, long typeId)	**long**	Returns the number of tasks assigned to a user.

Filter parameters description:

- The statusId parameter must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameter must be a valid type id.

Example:

```
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.getAssignedCount(statusId, projectId, typeId);
         System.out.println(total);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

getActiveCount

Description:

Method	Return values	Description
getActiveCount(long statusId, long projectId, long typeId)	**Long**	Returns the number of active tasks for a user.

Filter parameters description:

- The statusId parameter must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameter must be a valid type id.

Example:

```
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.getActiveCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getFinishedCount

Description:

Method	Return values	Description

getFinishedCount(long statusId, long projectId, long typeId)	**Long**	Returns the number of finished tasks for a user.

Filter parameters description:

- The statusId parameter must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameter must be a valid type id.

Example:

```
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.getFinishedCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotifiedCount

Description:

Method	Return values	Description
getNotifiedCount(long statusId, long projectId, long typeId)	**Long**	Returns the number of notified tasks assigned to a user.

Filter parameters description:

- The statusId parameter must be a valid status id.
- The projectId parameter must be a valid project id.
- The typeId parameter must be a valid type id.

Example:

```
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.getNotifiedCount(statusId, projectId, typeId);
            System.out.println(total);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

create

Description:

Method
create(String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> 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 the two day fields (day of month, or day of week) match the current time. The scheduler examines crontab entries o

minute. The time and date fields are:

```
* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ????? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the sam
? ? ? ?????????? 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.create(subject, start, end, description, statusId, projectId, typeId,
                  "okmAdmin", notificationUsers, externalUsers, relatedDocuments, relatedFolders, relatedRecords, relat
                  "dd-MM-yyyy", 10, "m", 5, "m", 10);
            System.out.println(newTask);
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   }
```

update

Description:

Method	Return values
update(long taskId, String subject, String start, String end, String description, long statusId, long projectId, long typeId, String user, List<String> notificationUsers, List<String> externalUsers, List<String> relatedDocuments, List<String> relatedFolders, List<String> relatedRecords, List<String> relatedMails, String owner, String repeatExpression, String repeatUntil, String formatDate, int repeatTimes, int progress, String reminderStartUnit, int reminderStartValue, String reminderEndUnit, int reminderEndValue)	**Task**

The **repeatExpression** parameters description:

The commands are executed by cron when the minute, hour, and month fields match the current time, and when at l
the two day fields (day of month, or day of week) match the current time. The scheduler examines crontab entries o

minute. The time and date fields are:

```
* * * * * command to execute
? ? ? ? ?
? ? ? ? ?
? ? ? ? ????? day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the sam
? ? ? ?????????? 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.update(taskId, subject, start, end, description,
                  statusId, projectId, typeId, "okmAdmin", notificationUsers, externalUsers,
                  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();
        }
    }
}
```

delete

Description:

Method	Return values	Description
delete(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.delete(taskId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createProject

Description:

Method	Return values	Description
createProject(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.createProject("Project one", true, "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateProject

Description:

Method	Return values	Description
updateProject(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.updateProject(projectId, false, "cancelled", "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteProject

Description:

Method	Return values	Description
deleteTProject(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.deleteProject(projectId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProject

Description:

Method	Return values	Description
getProject(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.getProject(projectId);
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getProjects

Description:

Method	Return values	Description
getProjects(boolean filterActive)	**List<TaskProject>**	Retrieve a list of all 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.getProjects(true)) {
                System.out.println(tp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

createType

Description:

Method	Return values	Description
createType(String name, boolean active, String description)	**TaskType**	Create a new task type.

The boolean parameter "active" indicates whether 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.createType("Type one", true, "Description");
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateType

Description:

Method	Return values	Description
updateType(long typeId, boolean active, String name, String description)	**TaskType**	Update a task type.

The boolean parameter "active" indicates whether 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.updateType(typeId, false, "Type one", "Description");
            System.out.println(tp);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteType

Description:

Method	Return values	Description
deleteType(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.deleteType(typeId);
```

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

getType

Description:

Method	Return values	Description
getType(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.getType(typeId);
            System.out.println(tt);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getTypes

Description:

Method	Return values	Description
getTypes(boolean filterActive)	**List<TaskType>**	Retrieve a list of all 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.getTypes(true)) {
                System.out.println(tt);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createStatus

Description:

Method	Return values	Description
createStatus(String name, boolean finish)	**TaskStatus**	Create a new task status.

Depending on your logic, several statuses can be ending statuses. For example, statuses named "closed" or "cancelled". The boolean parameter "finish" indicates whether your status is 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.createStatus("cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
}
```

updateStatus

Description:

Method	Return values	Description
updateStatus(long statusId, String name, boolean finish)	**TaskStatus**	Update a task status.

Depending on your logic, several statuses can be ending statuses. For example, statuses named "closed" or "cancelled". The boolean parameter "finish" indicates whether your status is 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.updateStatus(statusId, "cancelled", true);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteStatus

Description:

Method	Return values	Description
deleteStatus(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.deleteStatus(statusId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getStatus

Description:

Method	Return values	Description
getStatus(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.getStatus(statusId);
            System.out.println(ts);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
}
```

getStatus

Description:

Method	Return values	Description
getStatus()	**List<TaskStatus>**	Retrieve a list of all task statuses.

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.getStatus(null)) {
                System.out.println(ts);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

createNote

Description:

Method	Return values	Description
createNote(long taskId, String text)	**TaskNote**	Create a new note for 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.createNote(taskId, "New note");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

updateNote

Description:

Method	Return values	Description
updateNote(long noteId, String text)	**TaskNote**	Update a note for 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.updateNote(noteId, "Note updated");
            System.out.println(tn);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteNote

Description:

Method	Return values	Description
deleteNote(String token, long noteId)	**void**	Delete a note from 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.deleteNote(noteId);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

getNotes

Description:

Method	Return values	Description
getNotes(long taskId)	**List<TaskNote>**	Retrieve a list of all notes for 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.getNotes(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 you should log in using the method "**login**". You can access the "**login**" method from the webservice object "**ws**" as shown below:

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

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

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

```
ws.userConfig.getConfig();
```

Methods

getConfig

Description:

Method	Return values	Description
getConfig()	**UserConfig**	Returns the user settings.

Example:

```
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**	Sets the user's 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 web service object:

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

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

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

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

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

```
ws.wizard.findByUser();
```

Methods

findByUser

Description:

Method	Return values	Description
findByUser()	**List<WizardNode>**	Returns a list of nodes with a wizard.

Example:

```
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();
        }
    }
}
```

findAllByUser

Description:

Method	Return values	Description
findAllByUser()	**List<WizardNode>**	Returns a list of all nodes with an assigned 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.findAllByUser();
            for (WizardNode wizardNode : list) {
                System.out.println(wizardNode);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

findByUserAndUuid

Description:

Method	Return values	Description
findByUserAndUuid(String uuid)	**WizardNode**	Gets 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-4c12-8816-650b2c82c89b");
            System.out.println(wizardNode);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

hasWizardByUserAndNode

Description:

Method	Return values	Description
hasWizardByUserAndNode()	**boolean**	Checks 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.hasWizardByUserAndNode("11225723-883f-4c12-
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteAll

Description:

Method	Return values	Description
deleteAll()	**void**	Removes all wizards for the user.

Only wizards assigned to the user session will be removed.

Example:

```
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();
        }
    }
}
```

postponeNode

Description:

Return

Method	values	Description
postponeNode(String uuid, Calendar date)	**void**	Postpones the node with the specified UUID to the given date.

Example:

```
package com.openkm;

import com.openkm.sdk4j.OKMWebservicesFactory;
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);
            Calendar calendar = Calendar.getInstance();
            calendar.set(Calendar.DAY_OF_MONTH, 15);
            ws.wizard.postponeNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba", calendar);
            System.out.println("postponeNode");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

cancelPostponeNode

Description:

Method	Return values	Description
cancelPostponeNode(String uuid)	**void**	Cancels the postponement of the node with the specified 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.wizard.cancelPostponedNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
            System.out.println("cancelPostponedNode");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

deleteByNode

Description:

Method	Return values	Description
deleteByNode(String uuid)	**void**	Deletes the wizard 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.deleteByNode("cd4f69ed-e517-408b-b7eb-95cfe371ecba");
            System.out.println("deleteByNode");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addDigitalSignature

Description:

Method	Return values	Description
addDigitalSignature()	**void**	Adds a digital signature to 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**	Removes a digital signature from 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**	Adds show wizard categories to 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**	Removes show wizard categories from 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-922958f5b6ae");
            System.out.println("removeShowWizardCategories");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addShowWizardKeywords

Description:

Method	Return values	Description
addShowWizardKeywords	**void**	Adds show wizard keywords to 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**	Removes show wizard keywords from 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**	Adds show wizard OCR data capture to 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-922958f5b6ae");
            System.out.println("addShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

removeShowWizardOCRDataCapture

Description:

Method	Return values	Description
removeShowWizardOCRDataCapture	**void**	Removes show wizard OCR data capture from 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-922958f5b6ae");
            System.out.println("removeShowWizardOCRDataCapture");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

addGroup

Description:

Method	Return values	Description
addGroup(String uuid, String group)	**void**	Adds a metadata group to 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**	Removes a metadata group from 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**	Adds a workflow to 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**	Removes a workflow from 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", "purchase");
            System.out.println("addGroup");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

setAutostart

Description:

Method	Return values	Description
setAutostart	**void**	Sets auto start for the wizard.

Autostart should always be set as the last wizard option.

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 web service object:

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

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

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

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

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

```
ws.workflow.registerProcessDefinition(is);
```

For most examples, the Purchase workflow sample has been used.

Methods

registerProcessDefinition

Description:

Method	Return values	Description
registerProcessDefinition(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 is a valid workflow process definition 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 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 is a valid workflow process definition id.		

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 id. The parameter uuid can be any document, mail, folder or record UUID. The propertiesMap values are the form element values needed to start the workflow (not all workflows need form values to start).		

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-b04cea9c0048", properties);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

runProcessDefinition

Description:

Method	Return values	Description
runProcessDefinition(long pdId, Map<String, String> propertiesMap)	**ProcessInstance**	Executes a workflow without associating it to a node.
The parameter pdId value is a valid workflow process definition id. The propertiesMap values are the form element values needed to start the workflow (not all workflows need form values to start).		

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

findProcessInstances

Description:

Method	Return values	Description
findProcessInstances(long pdId)	**List<ProcessInstance>**	Retrieves a list of all process instances for some registered workflow definitions.
The parameter pdId is a valid workflow process definition id.		

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 workflow 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 latest workflow 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 the last workflow definition of a specific workflow.

The parameter name identifies a specific workflow definition group.

Several workflow definition versions 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 for 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**	Sets task instance values.

The parameter tiId is a valid task instance id.

The parameter transName is the chosen transaction.

The propertiesMap values are the 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", properties);
        } 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**	Sets the actor for a task instance.
The parameter tiId is a valid task instance id. The parameter actorId must be a valid OpenKM user 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.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>>**	Returns a map with all the process definition forms.
The parameter pdId value is a valid workflow process definition id.		

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.getProcessDefinitionForms(12);
            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 is a valid workflow process definition 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);
            String tiId = "1";
            String res = ws.workflow.getProcessDefinitionImage(tiId, "82ed9618-11eb-4b1f-87e7-f853ee60c3c0");
            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 for some registered workflow definitions.
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-48f1-8505-e83b26716956",
        } 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) of suggestBox values.

Example:

```
package com.openkm;

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

import java.util.Map;
```

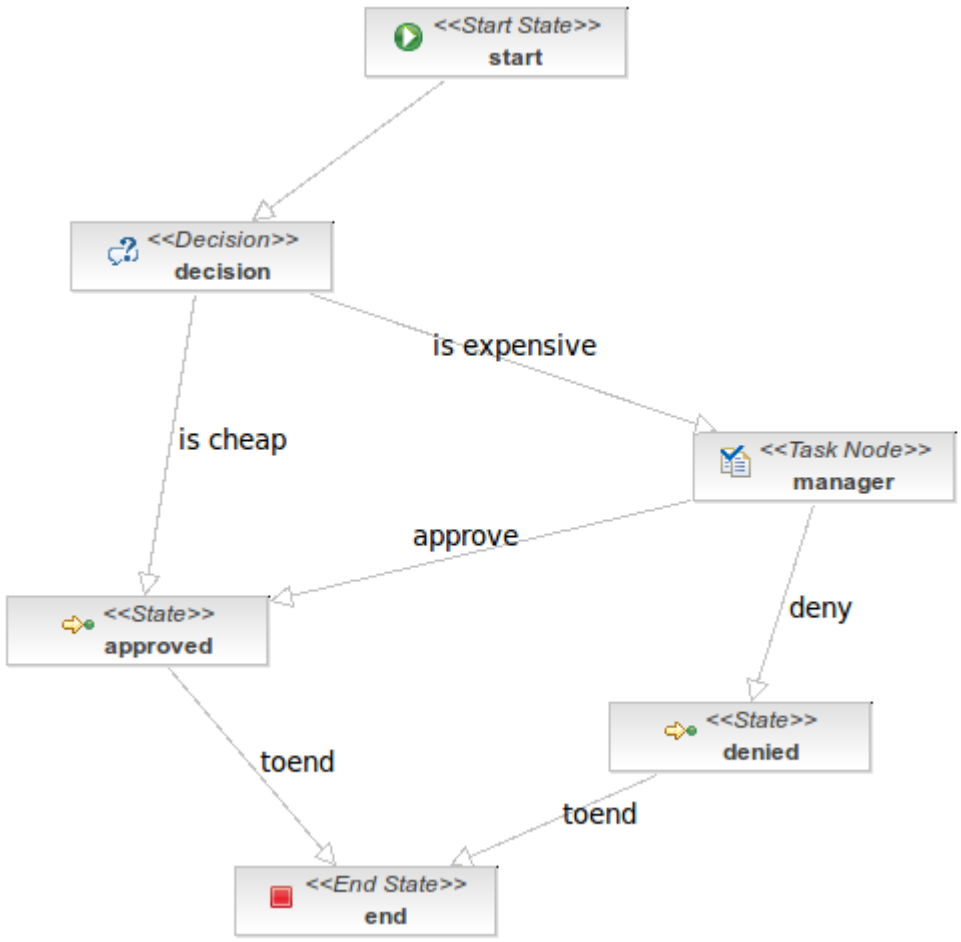
```
public class Test {

    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, "301de803-f4ae-48f1-8505-e
            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 a user called "manager" so you need to create this user and log in 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(&amp;quot;From APPROVED Go to END&amp;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(&amp;quot;From DENIED Go to END&amp;quot;);</script>
    </timer>
    <transition to="end" name="toend"></transition>
  </state>

  <end-state name="end"></end-state>
</process-definition>
```

Process handlers

None.

Form definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"
                               "http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="run_config">
    <input label="Purchase price" name="price" />
    <textarea label="Purchase description" name="description" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="evaluate price">
    <input label="Purchase price" name="price" data="price" readonly="true" />
    <textarea label="Purchase description" name="description" data="description" readonly="true" />
    <button name="approve" label="Approve" transition="approve"/>
    <button name="deny" label="Deny" transition="deny"/>
  </workflow-form>
</workflow-forms>
```