
# Documentation for SDK for .NET 2.0-ce

# Table of Contents

# SDK for .NET 2.0-ce

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

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

## License

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

This program is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the EULA - OpenKM SDK End User License Agreement for more details.

## Compatibility

**SDK for .NET version 2.0-ce should be used:**

- **From OpenKM Professional version 7.0.1.**

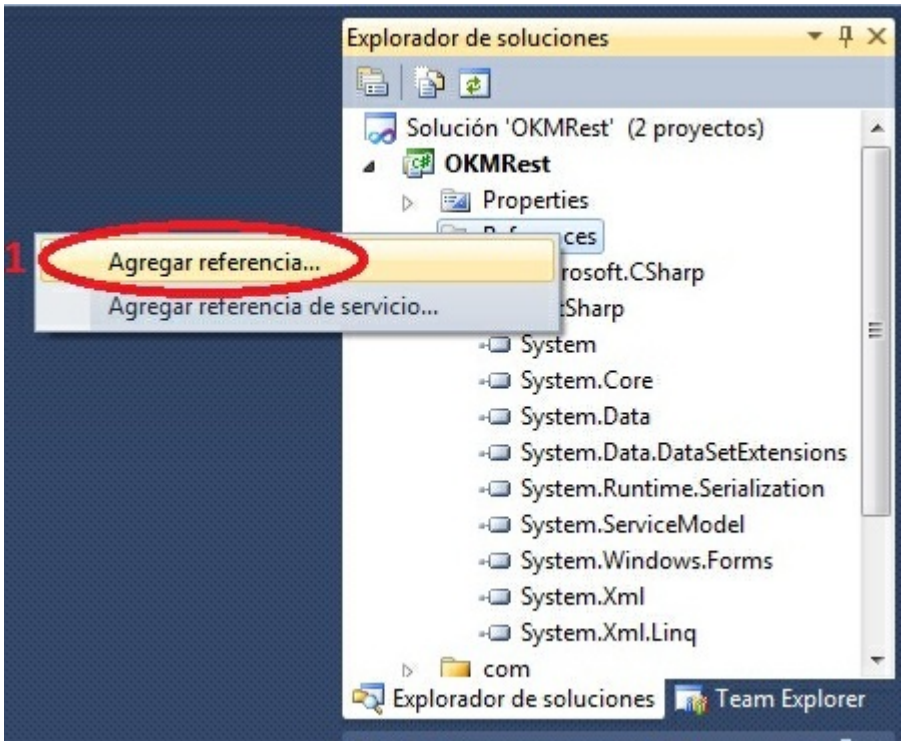
# Requirements

- **NetFramework 4.8.0**
- **Newtonsoft.Json.dll version 12.0.1.**
- **RestSharp.dll library version 106.10.1.**

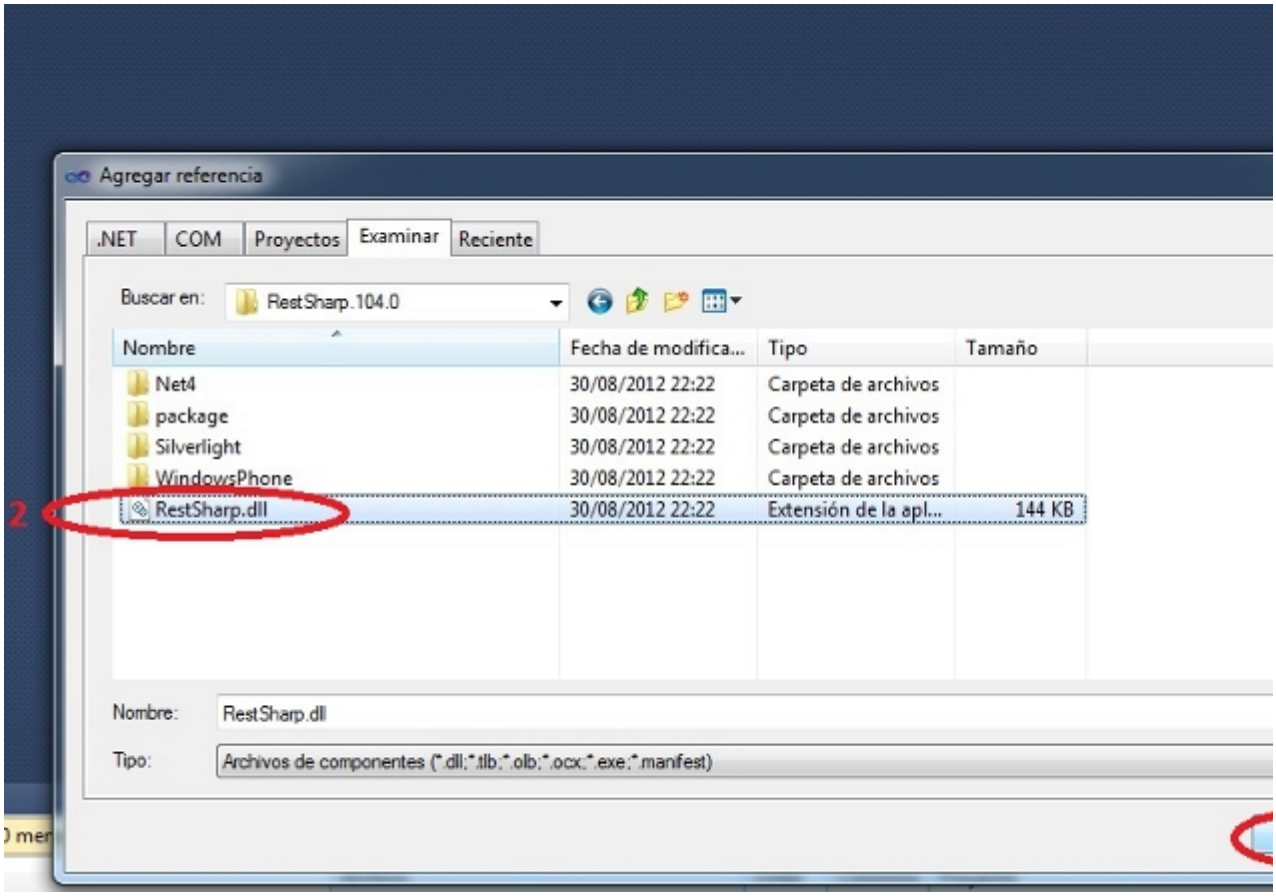
More information about RestSharp at http://restsharp.org/.

## Visual studio project configuration

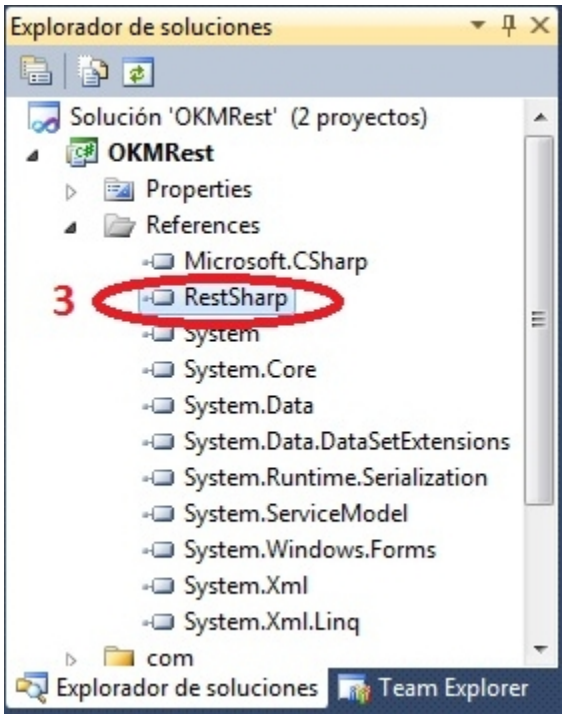
- Download the "**RestSharp.dll**" from here or **Manage NuGet Packages.**
- Right click on "**References**" and in the contextual menu choose "**add reference**".




- Choose the "**RestSharp.dll**" from your file system.




- Click on "**Accept**" button.



The dll library is yet configured in your .NET project.

# Sample client

Your first class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    class Program
    {
        /**
        * Sample OpenKM JDK client
        */
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach(Folder fld in  ws.folder.getFolderChildren("14530e37-ac51-4a26-8049-0146a5916dec"))
                {
                    System.Console.WriteLine("Folder -> " + fld.path);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }

            System.Console.ReadKey();
        }
    }
}
```

# Basic concepts

## Authentication

First, you must create the webservice object:

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

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

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

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

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

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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.exception;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getAppVersion().getVersion());
            } catch (RepositoryException e) {
                System.Console.WriteLine(e.ToString());
            } catch (DatabaseException e) {
                System.Console.WriteLine(e.ToString());
            } catch (UnknowException e) {
                System.Console.WriteLine(e.ToString());
            } catch (WebserviceException e) {
                System.Console.WriteLine(e.ToString());
            }
```

```
            System.Console.ReadKey();
        }
    }
}
```

## Accessing API

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

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

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

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

OpenKM API classes:

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

| | | | | |
|---|---|---|---|---|
| | delete a folder. | | | |
| **OKMMail** | Manages mails. For example create, move or delete a mails. | Yes | MailImpl.cs | BaseMail.cs |
| **OKMNode** | Manages general purpose node actions. | Yes | NodeImpl.cs | BaseNode.cs |
| **OKMNote** | Manages notes on any node type. For example create, edit or delete a note on a document, folder, mail or record. | Yes | NoteImpl.cs | BaseNote.cs |
| **OKMNotification** | Manages notifications. For example add or remove subscriptions on a document or a folder. | Yes | NotificationImpl.cs | BaseNotification.cs |
| **OKMProperty** | Manages categories and keywords. For example add or remove keywords on a document, folder, mail or record. | Yes | PropertyImpl.cs | BaseProperty.cs |
| **OKMPropertyGroup** | Manages metadata. For example add metadata group, set metadata fields. | Yes | PropertyGroupImpl.cs | BasePropertyGroup.cs |
| **OKMRecord** | Manages records. For example create, move or delete a record. | Yes | RecordImpl.cs | BaseRecord.cs |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **OKMRelation** | Manages relations between nodes. For example create a relation ( parent-child ) between two documents. | Yes | RelationImpl.cs | BaseRelation.cs |
| **OKMRepository** | A lot of material related with repository. For example get the properties of main root node ( /okm:root ). | Yes | RepositoryImpl.cs | BaseRepository.cs |
| **OKMReport** | Manages reports. For example execute reports. | Yes | ReportImpl.cs | BaseReport.cs |
| **OKMSearch** | Manages search feature. For example manage saved queries or perform a new query to the repository. | Yes | SearchImpl.cs | BaseSearch.cs |
| **OKMUserConfig** | Manages user home configuration. | No | UserConfigImpl.cs | BaseUserConfig.cs |

(*) Indicate that exist REST support for it, but still not implemented into the SDK.

## Class Hierarchy

Packages detail:

| Name | Description |
|---|---|
| **com.openkm.sdk4csharp** | The **com.openkm.OKMWebservicesFactory** that returns an **com.openkm.OKMWebservices** object which implements all interfaces.<br>`OKMWebservices ws = OKMWebservicesFactory.newInstance(host, username, password);` |

| | |
|---|---|
| **com.openkm.sdk4csharp.bean** | Contains all classes result of unmarshalling REST objects. |
| **com.openkm.sdk4csharp.definition** | All interface classes:<br>• com.openkm.sdk4csharp.definition.BaseActivity<br>• com.openkm.sdk4csharp.definition.BaseAuth<br>• com.openkm.sdk4csharp.definition.Bookmark<br>• com.openkm.sdk4csharp.definition.BaseConversion<br>• com.openkm.sdk4csharp.definition.BaseDashboard<br>• com.openkm.sdk4csharp.definition.BaseDocument<br>• com.openkm.sdk4csharp.definition.BaseFolder<br>• com.openkm.sdk4csharp.definition.BaseMail<br>• com.openkm.sdk4csharp.definition.BaseNode<br>• com.openkm.sdk4csharp.definition.BaseNote<br>• com.openkm.sdk4csharp.definition.BaseNotification<br>• com.openkm.sdk4csharp.definition.BaseProperty<br>• com.openkm.sdk4csharp.definition.BasePropertyGroup<br>• com.openkm.sdk4csharp.definition.BaseRecord<br>• com.openkm.sdk4csharp.definition.BaseRelation<br>• com.openkm.sdk4csharp.definition.BaseReport<br>• com.openkm.sdk4csharp.definition.BaseRepository<br>• com.openkm.sdk4csharp.definition.BaseSearch<br>• com.openkm.sdk4csharp.definition.BaseUserConfig<br>• com.openkm.sdk4csharp.definition.BaseWizard |
| **com.openkm.sdk4csharp.impl** | All interface implementation classes:<br>• com.openkm.sdk4csharp.impl.ActivityImpl<br>• com.openkm.sdk4csharp.impl.AuthImpl<br>• com.openkm.sdk4csharp.impl.BookmarkImpl<br>• com.openkm.sdk4csharp.impl.ConversionImpl<br>• com.openkm.sdk4csharp.impl.DashboardImpl<br>• com.openkm.sdk4csharp.impl.DocumentImpl<br>• com.openkm.sdk4csharp.impl.FolderImpl<br>• com.openkm.sdk4csharp.impl.ImportImpl |

| | |
|---|---|
| | • com.openkm.sdk4csharp.impl.MailImpl<br>• com.openkm.sdk4csharp.impl.NodeImpl<br>• com.openkm.sdk4csharp.impl.NoteImpl<br>• com.openkm.sdk4csharp.impl.NotificationImpl<br>• com.openkm.sdk4csharp.impl.PropertyGroupImpl<br>• com.openkm.sdk4csharp.impl.PropertyImpl<br>• com.openkm.sdk4csharp.impl.RecordImpl<br>• com.openkm.sdk4csharp.impl.RelationImpl<br>• com.openkm.sdk4csharp.impl.RepositoryImpl<br>• com.openkm.sdk4csharp.impl.SearchImpl<br>• com.openkm.sdk4csharp.impl.UserConfigImpl<br>• com.openkm.sdk4csharp.impl.WizardImpl |
| **com.openkm.sdk4csharp.util** | A couple of utilities.<br>The class com.openkm.sdk4csharp.util.ISO8601 should be used to set and parse metadata date fields. |
| **com.openkm.sdk4csharp.exception** | All exception classes. |

# Activity samples

## Basic

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**findActivityLog**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DateTime beginDate = new DateTime(2019, 7, 1);
            DateTime endDate = new DateTime(2019, 8, 1);
            ActivityList activityList = ws.activity.findActivityLog(0, 10, beginDate, endDate, "testUser", "LOGIN", null);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getActivityActions**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
```

```
            {
                ws.login(user, password);
                List<string> actions = ws.activity.getActivityActions();
                foreach(string action in actions)
                {
                    System.Console.WriteLine(action);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Auth samples

## Basics

The class **com.openkm.sdk4csharp.bean.Permission** contains permission values ( READ, WRITE, etc. ). You should use it in combination with methods that are changing or getting security grants.

To set READ and WRITE access you should do:

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

To check if you have permission access you should do:

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

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

Example of nodeId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**login**

Description:

| Method | Return values | Description |
|---|---|---|
| **login(String user, String password)** | **String** | Return the login authorized token. |

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

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

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

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**login**

Description:

| Method | Return values | Description |
|---|---|---|
| **String login(String user, String password, int expiration, bool restrictIp)** | String | Login process return authentication token. |

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

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

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

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

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            int days = 7;
            bool restrictIp = true;
            try
            {
                System.Console.WriteLine("Token: " + ws.login(user, password, days, restrictIp));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**logout**

Description:

| Method | Return values | Description |
|---|---|---|
| **logout()** | **void** | Execute the logout method in OpenKM side. |

The authentication token will be reset.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.logout();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getGrantedRoles**

Description:

| Method | Return values | Description |
|---|---|---|
| **getGrantedRoles(String nodeId)** | **Dictionary<String, int>** | Return the granted roles of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
         try
         {
            ws.login(user, password);
            Dictionary<String, int> grants = ws.auth.getGrantedRoles("14530e37-ac51-4a26-8049-0146a5916dec");
            foreach (String role in grants.Keys)
            {
                System.Console.WriteLine("role ->" + role);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getGrantedUsers**

Description:

| Method | Return values | Description |
|---|---|---|
| **getGrantedUsers(String nodeId)** | **Dictionary<String, int>** | Returns the granted users of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<String, int> grants = ws.auth.getGrantedUsers("14530e37-ac51-4a26-8049-0146a5916dec");
            foreach (KeyValuePair<string, int> kvp in grants)
            {
               Console.WriteLine("{0} -> {1}", kvp.Key, kvp.Value);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getRoles**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               foreach (String role in ws.auth.getRoles(false))
               {
                  System.Console.WriteLine(role);
               }
            } catch (Exception e) {
               System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getRolesByUser**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               foreach(String role in ws.auth.getRolesByUser("okmAdmin"))
               {
                  System.Console.WriteLine(role);
               }
            } catch (Exception e) {
               System.Console.WriteLine(e.ToString());
            }
        }
    }
```

```
}
```

**getUser**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUser(String userId)** | **CommonUser** | Returns a user. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            CommonUser user = ws.auth.getUser("okmAdmin");
            System.Console.WriteLine(user.toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUsers**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUsers(boolean showAll)** | **List<CommonUser>** | Returns the list of all the users. |
| When showAll variable is set to true return all the users - enabled and disabled - otherwise only returns the enabled. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (CommonUser user in ws.auth.getUsers(false))
            {
               System.Console.WriteLine(user.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUsersByRole**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (CommonUser user in ws.auth.getUsersByRole("ROLE_ADMIN"))
                {
                    System.Console.WriteLine(user.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**revokeRole**

Description:

| Method | Return values | Description |
|---|---|---|
| **revokeRole(String nodeId, String role, int permissions, boolean recursive)** | **void** | Removes a role grant on a node. |
| The parameter recursive only has a sense when the nodeId is a folder or record node.<br><br>When the parameter recursive is true, the change will be applied to the node and descendants. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
```

```
            ws.login(user, password);
            // Remove ROLE_USER write grants at the node but not descendants
            ws.auth.revokeRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_USER", Permission.ALL_GRAN

            // Remove all ROLE_ADMIN grants to the node and descendants
            ws.auth.revokeRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_ADMIN", Permission.ALL_GRA
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**revokeUser**

Description:

| Method | Return values | Description |
|---|---|---|
| **revokeUser(String nodeId, String user, int permissions, boolean recursive)** | **void** | Removes a user grant on a node. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // Remove john write grants at the node but not descendants
            ws.auth.revokeUser("14530e37-ac51-4a26-8049-0146a5916dec", "john", Permission.ALL_GRANTS, false

            // Remove all okmAdmin grants at the node and descendants
            ws.auth.revokeUser("14530e37-ac51-4a26-8049-0146a5916dec", "okmAdmin", Permission.ALL_GRANT
```

```
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**grantRole**

Description:

| Method | Return values | Description |
|---|---|---|
| **grantRole(String nodeId, String role, int permissions, boolean recursive)** | **void** | Add role grant on a node. |
| The parameter recursive only makes any sense when the nodeId is a folder or record node.<br><br>When the parameter recursive is true, the change will be applied to the node and descendants. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Add ROLE_USER write grants at the node but not descendants
                ws.auth.grantRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_USER", Permission.ALL_GRAN

                // Add all ROLE_ADMIN grants to the node and descendants
                ws.auth.grantRole("14530e37-ac51-4a26-8049-0146a5916dec", "ROLE_ADMIN", Permission.ALL_GRA
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**grantUser**

Description:

| Method | Return values | Description |
|---|---|---|
| **grantUser(String nodeId, String user, int permissions, boolean recursive)** | **void** | Adds a user grant on a node. |
| The parameter recursive only makes any sense when the nodeId is a folder or record node.<br><br>When the parameter recursive is true, the change will be applied to the node and descendants. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Add john write grants at the node but not descendants
                ws.auth.grantUser("14530e37-ac51-4a26-8049-0146a5916dec", "john", Permission.ALL_GRANTS, false)

                // Add all okmAdmin grants at the node and descendants
                ws.auth.grantUser("14530e37-ac51-4a26-8049-0146a5916dec", "okmAdmin", Permission.ALL_GRANTS,
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getProfiles**

Description:

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

| | | |
|---|---|---|
| **getProfiles(boolean filterByActive)** | **List<Profile>** | Return the list of all profiles. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (Profile profile in ws.auth.getProfiles(true))
            {
               System.Console.WriteLine(profile.toString());
            }
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserProfile**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUserProfile(String userId)** | **Profile** | Returns the profile assigned to a user. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.auth.getUserProfile("okmAdmin").toString());
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setUserProfile**

Description:

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
```

```
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Set the profile named "default" to the user
                foreach (Profile profile in ws.auth.getProfiles(true))
                {
                    if (profile.name.Equals("Default"))
                    {
                        ws.setUserProfile("okmAdmin", profile.id);
                    }
                }
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**createUser**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            ws.auth.createUser("test", "password.2016", "some@mail.com", "User Name", true);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**deleteUser**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**updateUser**

Description:

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

| | | |
|---|---|---|
| **updateUser(String user, String password, String email, String name, Boolean active)** | **void** | Update a user. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**createRole**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**deleteRole**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**updateRole**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**assingRole**

Description:

| Method | Return values | Description |
|---|---|---|
| **assingRole(String user, String role)** | **void** | Assign a role to a user. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

**removeRole**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**changeSecurity**

Description:

| Method | Return values | Description |
|---|---|---|
| **changeSecurity(ChangeSecurity changeSecurity)** | **void** | Change the security of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Folder fld = ws.folder.createFolder("a978f8e6-aa87-4f0f-b7bc-6f44cb090fdf", "Folder1");
            ChangeSecurity cs = new ChangeSecurity();
            List<GrantedRole> grList = new List<GrantedRole>();
            GrantedRole gr = new GrantedRole();
            gr.role = "ROLE_TEST";
            gr.permissions = Permission.READ | Permission.WRITE;
            grList.Add(gr);
            cs.grantedRolesList = grList;
            ws.auth.changeSecurity(fld.uuid, cs);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**overwriteSecurity**

Description:

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

The ChangeSecurity object is used in changeSecurity and overwriteSecurity methods.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            GrantedUser gu = new GrantedUser();
            gu.user = pherrera;
            gu.permissions = Permission.READ | Permission.WRITE;

            List<GrantedUser> guList = new List<GrantedUser>();
            guList.Add(gu);

            GrantedRole gr = new GrantedRole();
            gr.role = "ROLE_TEST";
            gr.permissions = Permission.READ | Permission.WRITE;

            List<GrantedRole> grList = new List<GrantedRole>();
            grList.Add(gr);

            ChangeSecurity changeSecurity = new ChangeSecurity();
            changeSecurity.recursive = false;
            changeSecurity.grantedUsersList = guList;
            changeSecurity.grantedRolesList = grList;

            ws.auth.overwriteSecurity("4f873d10-654e-4d99-a94f-15466e30a0f6", changeSecurity);
         }
         catch (Exception e)
         {
```

```
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getGrantedUsersAndRoles**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                GrantedUsersAndRolesItem grants = ws.auth.getGrantedUsersAndRoles("15e35aef-f1bb-451e-ac86-3b
                foreach (KeyValuePair<string, int> user in grants.grantedUsers)
                {
                    System.Console.WriteLine(user.Key + "->" + user.Value);
                }
                foreach (KeyValuePair<string, int> role in grants.grantedRoles)
                {
                    System.Console.WriteLine(role.Key + "->" + role.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setUserPermissions**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Set user permissions
                ws.auth.setUserPermissions("22c1d190-f798-489d-b420-2008cb38705b", "user1", Permission.READ + P

                // Update permissions of okmAdmin at the node and descendants
                ws.auth.setUserPermissions("22c1d190-f798-489d-b420-2008cb38705b", "okmAdmin", Permission.ALL_
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setRolePermissions**

Description:

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

| | | |
|---|---|---|
| **setRolePermissions(String uuid, String roleId, int permissions, bool recursive)** | **void** | Update role permissions on a node. |
| The parameter recursive only has sense when the uuid is a folder or record node.<br><br>When parameter recursive is true, the change will be applied to the node and descendants. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // Set role permissions
            ws.auth.setRolePermissions("22c1d190-f798-489d-b420-2008cb38705b", "ROLE_USER", Permission.RE

            // Update permissions of ROLE_ADMIN at the node and descendants
            ws.auth.setRolePermissions("22c1d190-f798-489d-b420-2008cb38705b", "ROLE_ADMIN", Permission.A
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserTenants**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Tenant> tenants = ws.auth.getUserTenants();
                foreach (Tenant tenant in tenants)
                {
                    System.Console.WriteLine(tenant.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setUserTenant**

Description:

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

A user might have access to several tenants but only have one assigned.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long tenantId = 1; // Valid tenant id
                ws.auth.setUserTenant(tenantId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isAdmin**

Description:

| Method | Return values | Description |
|---|---|---|
| **isAdmin()** | **bool** | Check if the authenticated user is a member of administrators. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                bool isAdmin = ws.auth.isAdmin();
            }
            catch (Exception e)
            {
```

```
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getSessionId**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String id = ws.auth.getSessionId();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**hasSecurityRecursive**

Description:

| Method | Return values | Description |
|---|---|---|
| **hasSecurityRecursive()** | **bool** | Check if the user has grants to propagate the security recursively. |

The role is set in the configuration parameter named "**default.security.recursive.role**"

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                bool hasSecurityRecursive = ws.auth.hasSecurityRecursive();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isLoginLowercase**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.auth.isLoginLowercase().ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isPasswordExpired**

Description:

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



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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Password expired= " + ws.auth.isPasswordExpired().ToString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getUserToken**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUserToken(String userId)** | String | Return the auth token from a given user. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String userToken = ws.auth.getUserToken("testUser");
                Console.WriteLine("UserToken: " + userToken);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
```

```
    }
}
```

# Bookmark samples

## Basic

### Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

### getUserBookmarks

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Bookmark> bookmarks = ws.bookmark.getUserBookmarks();
                foreach (Bookmark bm in bookmarks)
                {
                    System.Console.WriteLine(bm.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**createBookmark**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.bookmark.createBookmark("a37f570f-5e7f-4a03-8d04-9b3689be82f1", "Any name");
            }
```

```
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**renameBookmark**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int bookmarkId = 1;
                ws.bookmark.renameBookmark(bookmarkId, "New name");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteBookmark**

Description:

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

| | | |
|---|---|---|
| **deleteBookmark(int bookmarkId)** | **void** | Delete a bookmark. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

# Conversion samples

## Basic

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**doc2pdf**

Description:

| Method | Return values | Description |
|---|---|---|
| **doc2pdf(FileStream is, String fileName)** | **Stream** | Retrieve the uploaded document converted to PDF format. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream inputFile = new FileStream("C:\\Documents\\test.docx", FileMode.Open);
            Stream tmpFile= ws.conversion.doc2pdf(inputFile, "test.docx");
            inputFile.Close();
            FileStream outputFile = new FileStream("C:\\Documents\\out.pdf", FileMode.OpenOrCreate, FileAccess.R
            tmpFIle.CopyTo(outputFile);
            outputFile.Close();
            tmpFIle.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**imageConvert**

Description:

| Method | Return values | Description |
|---|---|---|
| **imageConvert(FileStream fs, String fileName, List<String> params, String dstMimeType)** | **Stream** | Retrieve the uploaded image with transformation. |

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

The parameter dstMimeType is the expected document MIME TYPE result.

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

You can set a lot of 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 in OpenKM server to get it running.

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

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

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

Example:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fileStream = new FileStream("C:\\Documents\\test.jpg", FileMode.Open, FileAccess.Read);
            List<string> param = new List<string>();
            param.Add("-resize 50% ${fileIn} ${fileOut}");
            Stream tmpFIle = ws.conversion.imageConvert(fileStream, test.jpg, param, "image/png");
            fileStream.Close();
            FileStream destFile = new FileStream("C:\\Documents\\test.png", FileMode.OpenOrCreate);
            tmpFIle.CopyTo(destFile);
            destFile.Close();
            tmpFIle.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**html2pdf**

Description:

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

| | | |
|---|---|---|
| **html2pdf(String url)** | **Stream** | Retrieve the PDF of an URL. |

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

Example:

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Stream tmpFile = ws.conversion.html2pdf("http://www.openkm.com");
            FileStream destFile = new FileStream("C:\\Documents\\test.pdf", FileMode.OpenOrCreate);
            tmpFile.CopyTo(destFile);
            destFile.Close();
            tmpFile.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**doc2txt**

Description:

| Method | Return values | Description |
|---|---|---|
| **doc2txt(FileStream fs, String fileName)** | **String** | Extracts the text from the uploaded document. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fileStream = new FileStream("C:\\Documents\\test.docx", FileMode.Open);
            System.Console.WriteLine(ws.conversion.doc2txt(fileStream, "test.docx"));
            fileStream.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**img2txt**

Description:

| Method | Return values | Description |
|---|---|---|
| **img2txt(FileStream fs, String fileName)** | **String** | Extracts the text from the uploaded image. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("C:\\Images\\test.png", FileMode.Open);
                System.Console.WriteLine(ws.conversion.img2txt(fileStream, "test.png"));
                fileStream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**barcode2txt**

Description:

| Method | Return values | Description |
|---|---|---|
| **barcode2txt(FileStream fs, String fileName)** | **String** | Extracts the barcode text from the uploaded image. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            try
            {
                ws.login(user, password);
                FileStream fileStream = new FileStream("C:\\Images\\test.png", FileMode.Open);
                System.Console.WriteLine(ws.conversion.barcode2txt(fileStream, "test.png"));
                fileStream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Dashboard samples

## Basic

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

```
ws.dashboard.getUserCheckedOutDocuments();
```

## Methods

**getUserCheckedOutDocuments**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserCheckedOutDocuments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastModifiedDocuments**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastModifiedDocuments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLockedDocuments**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUserLockedDocuments(int offset, int limit)** | **DashboardResultSet** | Returns a list of the documents locked by the user. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedDocuments(0, 5);
```

```
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getUserLockedRecords**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
```

```
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLockedRecords(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getUserLockedFolders**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedFolders(0 ,5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLockedMails**

Description:

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

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

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

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

- limit=10

- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLockedMails(0 ,5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserSubscribedDocuments**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedDocuments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserSubscribedFolders**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserSubscribedFolders(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
```

```
                System.Console.WriteLine(dashboardResult.node.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getUserSubscribedRecords**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
```

```
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserSubscribedRecords(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getUserLastCreatedDocuments**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedDocuments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastDocumentsNotesCreated**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastDocumentsNotesCreated(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastCreatedFolders**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedFolders(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastFoldersNotesCreated**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastFoldersNotesCreated(0, 5);
```

```
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getUserLastCreatedRecords**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastCreatedRecords(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastRecordsNotesCreated**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastRecordsNotesCreated(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
                System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastDownloadedDocuments**

Description:

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

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

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

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

values:

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastDownloadedDocuments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserLastImportedMails**

Description:

| Method | Return values | Description |
|---|---|---|
| **getUserLastImportedMails(int offset, int** | **DashboardResultSet** | Returns a list of the last mails imported by the |

| | | |
|---|---|---|
| **limit)** | | user. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMails(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
```

```
    }
}
```

**getUserLastMailsNotesCreated**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            try
            {
                ws.login(user, password);
                DashboardResultSet resultSet = ws.dashboard.getUserLastMailsNotesCreated(0, 5);
                foreach (DashboardResult dashboardResult in resultSet.results)
                {
                    System.Console.WriteLine(dashboardResult.node.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getUserLastImportedMailAttachments**

Description:

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            DashboardResultSet resultSet = ws.dashboard.getUserLastImportedMailAttachments(0, 5);
            foreach (DashboardResult dashboardResult in resultSet.results)
            {
               System.Console.WriteLine(dashboardResult.node.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getUserSearches**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
```

```
            {
                ws.login(user, password);
                List<QueryParams> results = ws.dashboard.getUserSearches();
                foreach (QueryParams userSearch in results)
                {
                    System.Console.WriteLine(userSearch.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**findUserSearches**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<DashboardNodeResult> results = ws.dashboard.findUserSearches();
                foreach (DashboardNodeResult nodeResult in results)
                {
                    System.Console.WriteLine(nodeResult.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
```

```
    }
}
```

# Document samples

## Basics

On most methods, you'll see the parameter named "**docId**" and "**fldId**". The values of this parameter can be a valid document and folder **UUID** respectively.

Example of docId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**createDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **createDocument(String fldId, FileInfo fi)** | **Document** | Creates a new document and returns an object of type Document. |

**Parameters:**

- **fldId** is UUID of the folder where the document will be created.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileInfo fileInfo = new FileInfo("E:\\doc.docx");
            ws.document.createDocument(fldId, fileInfo);
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createDocument**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fileStream = new FileStream("E:\\logo.png", FileMode.Open);
            long nodeClass = 0;
            ws.document.createDocument("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43", "logo.png", fileStream, nodeC
            fileStream.Dispose();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **createDocument(String fldId, String name, FileStream fs)** | **Document** | Creates a new document and returns and object of type Document |
| **Parameters:**<br>• **fldId** is UUID of the folder where the document will be created.<br>• **name** is the name of the document including the extension. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fileStream = new FileStream("E:\\logo.png", FileMode.Open);
            ws.document.createDocument(fldId, "logo.png", fileStream);
            fileStream.Dispose();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteDocument(String docId)** | **void** | Deletes a document. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
```

```
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.deleteDocument("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }

            System.Console.ReadKey();
        }
    }
}
```

**getDocumentProperties**

Description:

| Method | Return values | Description |
|---|---|---|
| **getDocumentProperties(String docId)** | **Document** | Returns the document properties. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.getDocumentProperties("f123a950-0329-4d62-8328-0ff500fd42
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getContent**

Description:

| Method | Return values | Description |
|---|---|---|
| **getContent(String docId)** | **Stream** | Retrieves a document content - binary data - of the actual document version. |

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

```
Document doc = ws.getDocumentProperties(docId);
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

We've found wrong size problems while using:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Stream tmpFile = ws.document.getContent("f123a950-0329-4d62-8328-0ff500fd42db");
            FileStream destFile = new FileStream("C:\\test.png", FileMode.OpenOrCreate);
            tmpFile.CopyTo(destFile);
            destFile.Close();
            tmpFile.Close();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
```

```
        }
    }
}
```

**getContentByVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **getContentByVersion(String docId,String versionId)** | **Stream** | Retrieves a document content ( binary data ) of some specific document version. |

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

```
Document doc = ws.getDocumentProperties(docId);
response.setContentLength(new Long(doc.getActualVersion().getSize()).intValue());
```

We've found wrong size problems while using:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Stream tmpFile = ws.document.getContentByVersion("f123a950-0329-4d62-8328-0ff500fd42db","1.1");
                FileStream destFile = new FileStream("C:\\test.png", FileMode.OpenOrCreate);
                tmpFIle.CopyTo(destFile);
                destFile.Close();
```

```
            tmpFile.Close();
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getDocumentChildren**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
  public class Program
  {
    static void Main(string[] args)
    {
      String host = "http://localhost:8080/openkm";
      String username = "okmAdmin";
      String password = "admin";
      OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

      try
      {
        ws.login(user, password);
        foreach (Document doc in ws.document.getDocumentChildren("f123a950-0329-4d62-8328-0ff500fd42db"
        {
          System.Console.WriteLine(doc);
        }
      } catch (Exception e) {
        System.Console.WriteLine(e.ToString());
      }
    }
  }
}
```

**renameDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **renameDocument(String docId, String newName)** | **document** | Changes the name of a document. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Document doc = ws.document.renameDocument("f123a950-0329-4d62-8328-0ff500fd42db", "new_name
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setProperties**

Description:

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

Variables allowed to be changed:

- Title
- Description
- Language
- Associated categories

- Associated keywords

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<string> keywords = new List<string>();
            keywords.Add("testA");
            keywords.Add("testB");
            List<string> categories = new List<string>();
            categories.Add("/okm:categories");
            ws.document.setProperties("f123a950-0329-4d62-8328-0ff500fd42db", "anyTitle", "anyDescription", "es-E
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**checkout**

Description:

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

Only one user can modify a document at the same time.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.document.checkout("f123a950-0329-4d62-8328-0ff500fd42db");
            // At this point the document is locked for other users except for the user who executed the action
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**cancelCheckout**

Description:

| Method | Return values | Description |
|---|---|---|
| **cancelCheckout(String docId)** | **void** | Cancel a document edition. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(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("f123a950-0329-4d62-8328-0ff500fd42db");
                // At this point other users are allowed to execute a checkout and modify the document
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**forceCancelCheckout**

Description:

| Method | Return values | Description |
|---|---|---|
| **forceCancelCheckout(String docId)** | **void** | Cancels a document edition. |

This method allows canceling the edition on any document.

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(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("f123a950-0329-4d62-8328-0ff500fd42db");
                // At this point other users are allowed to execute a checkout and modify the document
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isCheckedOut**

Description:

| Method | Return values | Description |
|---|---|---|
| **isCheckedOut(String docId)** | **Boolean** | Returns a boolean that indicates if the document is being edited or not. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Is the document checkout:"+ ws.document.isCheckedOut("f123a950-0329-4d
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**checkin**

Description:

| Return |
|---|

| Method | values | Description |
|---|---|---|
| **checkin(String docId, String comment, FileStream fs)** | **Version** | Updates a document with a new version and returns an object with new Version values. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\logo.png", FileMode.Open);
            ws.document.checkin("f123a950-0329-4d62-8328-0ff500fd42db","optional some comment",fs);
            fs.Dispose();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**checkin**

Description:

| Method | Return values | Description |
|---|---|---|
| **checkin(String docId, String comment, FileStream fs, 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 edition - checkout - is allowed to update the document.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream(@"C:\Desktop\logo.png", FileMode.Open);
            ws.document.checkin("f123a950-0329-4d62-8328-0ff500fd42db", "optional some comment", fs, 1);
            fs.Dispose();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**isLocked**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine("Is document locked:" + ws.document.isLocked("f123a950-0329-4d62-8328-0f
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getLockInfo**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.document.getLockInfo("f123a950-0329-4d62-8328-0ff500fd42db"));
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
```

```
}
```

**purgeDocument**

Description:

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.document.purgeDocument("f123a950-0329-4d62-8328-0ff500fd42db");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**moveDocument**

Description:

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

| | | |
|---|---|---|
| **moveDocument(String docId, String dstId)** | **void** | Moves a document into a folder or record. |
| The values of the dstId parameter should be a folder or record UUID or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.document.moveDocument("f123a950-0329-4d62-8328-0ff500fd42db", fldId);
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**copyDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **copyDocument(String docId, String fldId, String newName)** | **Document** | Copies a document into a folder or record. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.document.copyDocument("f123a950-0329-4d62-8328-0ff500fd42db", fldId, "name.png");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getVersionHistorySize**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
```

```
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long byteCount = ws.document.getVersionHistorySize(docUUid);
                string[] suf = { "B", "KB", "MB", "GB", "TB", "PB", "EB" }; //Longs run out around EB
                long bytes = Math.Abs(byteCount);
                int place = Convert.ToInt32(Math.Floor(Math.Log(bytes, 1024)));
                double num = Math.Round(bytes / Math.Pow(1024, place), 1);
                System.Console.WriteLine((Math.Sign(byteCount) * num).ToString() + suf[place]);
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isValidDocument**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Return true
                ws.document.isValidDocument("f123a950-0329-4d62-8328-0ff500fd42db");

                // Return false
                ws.document.isValidDocument("/okm:root");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getDocumentPath**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.document.getDocumentPath("e339f14b-4d3a-489c-91d3-05e4575709d2"))
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**extendedDocumentCopy**

Description:

| Method | Return values | Description |
|---|---|---|
| **extendedDocumentCopy(String docId, String fldId, String name, bool categories, boolean keywords, bool propertyGroups, bool notes, bool wiki, bool security)** | **void** | Copies a document with associated data into a 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 parameter newName value is null, the document will preserve the same name.

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

Additional:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.document.extendedDocumentCopy("f123a950-0329-4d62-8328-0ff500fd42db", "d1cad1b0-3c4f-4ad3-
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getExtractedText**

Description:

| Method | Return values | Description |
|---|---|---|
| **getExtractedText(String docId)** | **String** | Returns a String with the extracted text by text extractor process. |

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

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

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

More information at Statistics.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.document.getExtractedText("f123a950-0329-4d62-8328-0ff500fd42db"));
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setExtractedText**

Description:

| Method | Return values | Description |
|---|---|---|
| **setExtractedText(String uuid, FileStream fileStream)** | **void** | Set extracted text |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream(@"C:\test.pdf", FileMode.Open, FileAccess.Read);
            ws.document.setExtractedText("f123a950-0329-4d62-8328-0ff500fd42db", fs);
            fs.Close();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getThumbnail**

Description:

| Method | Return values | Description |
|---|---|---|
| **getThumbnail(String docId, ThumbnailType type)** | **Stream** | Returns a 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Stream st = ws.document.getThumbnail("f123a950-0329-4d62-8328-0ff500fd42db", ThumbnailType.THUI
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setLanguage**

Description:

| Method | Return values | Description |
|---|---|---|
| **setLanguage(String docId, String lang)** | **void** | Sets a 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
```

```
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setLanguage("f123a950-0329-4d62-8328-0ff500fd42db", "es-ES");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setDocumentTitle**

Description:

| Method | Return values | Description |
|---|---|---|
| **setDocumentTitle(String docId, String title)** | **void** | Sets a document title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setDocumentTitle("f123a950-0329-4d62-8328-0ff500fd42db", "Some title here");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
```

```
}
```

**createDocumentFromTemplate**

Description:

| Method | Return values | Description |
|---|---|---|
| **createDocumentFromTemplate(String uuid, String dstPath , Boolean categories, Boolean keywords, Boolean notes, Boolean security, Dictionary<String, String> properties)** | **Document** | Creates a new document from a template and returns an object Document. |

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

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

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

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

Additional:

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

Example:

The example below is based on Creating PDF template sample.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
   public class Program
```

```
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new  Dictionary<String, String>();
                properties.Add("okp:tpl.name", "Some name");
                DateTime date = DateTime.Now;
                // Value must be converted to String ISO 8601 compliant
                properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
                properties.Add("okp:tpl.language", "java");
                ws.document.createFromTemplate("f123a950-0329-4d62-8328-0ff500fd42db", "/okm:root/test.pdf", false,
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**updateDocumentFromTemplate**

Description:

| Method | Return values | Description |
|---|---|---|
| **updateDocumentFromTemplate(String uuid, String dstId, Map<String, String> properties)** | **Document** | Updates a document previously created from a template and returns an object Document. |

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

The dstId is the document to updated UUID.

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

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

Example:

The example below is based on Creating PDF template sample.

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<String, String> properties = new  Dictionary<String, String>();
            properties.Add("okp:tpl.name", "Some name");
            DateTime date = DateTime.Now;
            // Value must be converted to String ISO 8601 compliant
            properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
            properties.Add("okp:tpl.language", "java");

            ws.document.updateFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "058b9379-b441-454d-85
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getDetectedLanguages**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (String lang in ws.document.getDetectedLanguages())
                {
                    System.Console.WriteLine(lang);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getCheckedOut**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Document> docs =  ws.document.getCheckedOut();
                foreach (Document doc in docs)
                {
                    System.Console.WriteLine(doc.toString());
                }
            }
            catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setDocumentDescription**

Description:

| Method | Return values | Description |
|---|---|---|
| **setDocumentDescription(String docId, String description)** | **void** | Sets a document description. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.document.setDocumentDescription("f123a950-0329-4d62-8328-0ff500fd42db", "Some description here
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**createWizardDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **createWizardDocument(String docId, long stage)** | **WizardNode** | Create a document with wizard. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream(LOCAL_DOC, FileMode.Open, FileAccess.Read);
            long nodeClass= 1;
            WizardNode wizardNode = ws.document.createWizardDocument("f123a950-0329-4d62-8328-0ff500fd42c
            fs.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getNumberOfPages**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("The number of pages is : " + ws.document.getNumberOfPages("f123a950-03
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getPageAsImage**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.document.getPageAsImage("f123a950-0329-4d62-8328-0ff500fd42db", 1,
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
```

```
    }
}
```

**IsAttachment**

Description:

| Method | Return values | Description |
|---|---|---|
| **isAttachment(String docId)** | **Boolean** | Returns a boolean that indicates if the node is a mail attachment or not. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.Writeline(ws.document.isAttachment("1ec49da9-1746-4875-ae32-9281d7303a62").ToSt
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getDocumentPdf**

Description:

| Method | Return values | Description |
|---|---|---|
| **getDocumentPdf(String uuid)** | **Stream** | Returns a PDF of the document. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Document doc = ws.document.getDocumentProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
            Stream stream = ws.document.getDocumentPdf(doc.uuid);
            FileStream docFile = new FileStream("D:\\test.pdf", FileMode.OpenOrCreate, FileAccess.ReadWrite);
            stream.CopyTo(docFile);
            stream.Close();
            docFile.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

# Folder samples

## Basics

On most methods, you'll see the parameter named "**fldId**". The value of this parameter can be some valid folder **UUID**.

Example of fldId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**createFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **createFolder(String fldId, String name)** | **Folder** | Creates a new folder and returns, as a result, an object Folder. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.folder.createFolder("f123a950-0329-4d62-8328-0ff500fd42db", "test");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getFolderProperties**

Description:

| Method | Return values | Description |
|---|---|---|
| **getFolderProperties(String fldId)** | **Folder** | Returns the folder properties. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            System.Console.WriteLine(ws.folder.getFolderProperties("f123a950-0329-4d62-8328-0ff500fd42db").toSt
        } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**deleteFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteFolder(String fldId)** | **void** | Delete a folder. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.folder.deleteFolder("f123a950-0329-4d62-8328-0ff500fd42db");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**renameFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **renameFolder(String fldId, String newName)** | **Folder** | Renames a folder. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // Exists folder /okm:root/test
            ws.folder.renameFolder("f123a950-0329-4d62-8328-0ff500fd42db", "renamedFolder");
            // Folder has renamed to /okm:root/renamedFolder
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**moveFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **moveFolder(String fldId, String dstId)** | **void** | Moves a folder into some folder or record. |
| The values of the dstId parameter should be a folder or record UUID or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
```

```
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // Exists folder /okm:root/test
                ws.folder.moveFolder("f123a950-0329-4d62-8328-0ff500fd42db", "ac165cab-a62a-4b17-89fc-2480a1d78
                // Folder has moved to /okm:root/tmp/test
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getFolderChildren**

Description:

| Method | Return values | Description |
|---|---|---|
| **getFolderChildren(String fldId)** | **List<Folder>** | Returns a list of all folder which their parent is fldId. |
| The parameter fldId can be a folder or a record node. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Folder fld in ws.folder.getFolderChildren("f123a950-0329-4d62-8328-0ff500fd42db"))
                {
                    System.Console.WriteLine(fld.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
```

```
        }
      }
   }
}
```

**isValidFolder**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // Return true
            ws.folder.isValidFolder("f123a950-0329-4d62-8328-0ff500fd42db");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getFolderPath**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.folder.getFolderPath("f123a950-0329-4d62-8328-0ff500fd42db"));
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**copyFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **copyFolder(String fldId, String dstId, String newName)** | **Folder** | Copies a folder into a folder or record. |

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

When parameter newName 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 "**extendedFolderCopy**" method for this feature.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.folder.copyFolder("f123a950-0329-4d62-8328-0ff500fd42db", "a321a950-0329-4d62-8328-0ff500fd42c
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**extendedFolderCopy**

Description:

| Method | Return values | Description |
|---|---|---|
| **extendedFolderCopy(String fldId, String dstId, String newName, bool categories, bool keywords, bool propertyGroups, bool notes, bool wiki, bool security)** | **Folder** | Copies a folder with associated data into some folder or record. |

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

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

By default, only the binary data and the security grants, the metadata, keywords, etc. of the folder are not 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 property groups parameter is true the original values of the metadata groups will be copied.
- When the node parameter is true the original values of the notes will be copied.
- When wiki parameter is true the original values of the wiki will be copied.
- When the security parameter is true the original value of the security will be copied.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.folder.extendedFolderCopy("f123a950-0329-4d62-8328-0ff500fd42db", "055e4384-7c70-4456-b32b-f5
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getContentInfo**

Description:

| Method | Return values | Description |
|---|---|---|
| **getContentInfo(String fldId)** | **ContentInfo** | Returne an object ContentInfo with information about folder. |

The ContentInfo object retrieves information about:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.folder.getContentInfo(fldUUid).toString());
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**purgeFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **purgeFolder(String fldId)** | **void** | The folder is definitely removed from the repository. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.folder.purgeFolder("f123a950-0329-4d62-8328-0ff500fd42db");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**createMissingFolders**

Description:

| Method | Return values | Description |
|---|---|---|
| **createMissingFolders(String fldPath)** | void | Create missing folders. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               ws.folder.createMissingFolders("/okm:root/missingfld1/missingfld2/missingfld3");
            } catch (Exception e) {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setFolderDescription**

Description:

| Method | Return values | Description |
|---|---|---|
| **setFolderDescription(string fldId, String description)** | **void** | Set the folder description. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.folder.setFolderDescription("f123a950-0329-4d62-8328-0ff500fd42db","Any description");
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createFolderFromTemplate**

Description:

| Method | Return values | Description |
|---|---|---|
| **createFolderFromTemplate(String uuid, String dstPath, bool categories, bool keywords, bool notes, bool propertyGroups, bool security, Dictionary<String, String> properties)** | **Folder** | Creates a new folder from the template and returns an object Folder. |

The **uuid** parameter is the template file.

The **dstPath** can be a folder or a record path.

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

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

Additional:

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

Example:

The example below is based on Creating PDF template sample.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("okp:tpl.name", "Some name");
                DateTime date = DateTime.Now;
                // Value must be converted to String ISO 8601 compliant
                properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
                properties.Add("okp:tpl.language", "[ \"java\" ]");
                Folder foder = ws.folder.createFolderFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "/okm:ro
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Import samples

## Basic

We suggest executing the methods below for huge import or for simple creation cases.

These methods execute fewer steps in the background either what is described in Document samples and Folder samples. That means you will get extra performance in the action because there are executed less logic in the server side.

Example of uuid:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

```
ws.quickImport.importDocument("1be884f4-5758-4985-94d1-f18bfe004db8", fileInfo);
```

## Methods

**importDocument**

Description:

| Method | Return values | Description |
|---|---|---|
| **importDocument(String uuid, FileInfo fi)** | **String** | Creates a new document and return the UUID of the Document. |

| | | |
|---|---|---|
| | | |

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

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
          String host = "http://localhost:8080/openkm";
          String username = "okmAdmin";
          String password = "admin";
          OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

          try
          {
              ws.login(user, password);
              FileInfo fileInfo = new FileInfo("E:\\doc.docx");
              String uuid = ws.quickImport.importDocument("22c1d190-f798-489d-b420-2008cb38705b", fileInfo);
              System.Console.WriteLine("UUID ="+ uuid);
          }
          catch (Exception e)
          {
              System.Console.WriteLine(e.ToString());

          }
      }
  }
}
```

**importFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **importFolder(String uuid, String name)** | **String** | Creates a new folder and return the UUID of the folder. |

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

Example:

```
using System; using System.Collections.Generic;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String uuid = ws.quickImport.importFolder("22c1d190-f798-489d-b420-2008cb38705b", "Folder-Name");
                System.Console.WriteLine("UUID ="+ uuid);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());

            }
        }
    }
}
```

# Mail samples

## Basics

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

Example of fldId:

- Using UUID -> "**50b7a5b9-89d2-430e-bbc9-6a6e01662a71**"

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**getMailProperties**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailProperties(String mailId)** | **Mail** | Returns the mail properties. |

Example:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.mail.getMailProperties("f0064cf3-4776-44c2-868a-f08697a6957e").toString
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteMail(String mailId)** | **void** | Deletes a mail. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.deleteMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
         }
```

```
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**purgeMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **purgeMail(String mailId)** | **void** | Folder is definitely removed from repository. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.purgeMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**renameMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **renameMail(String mailId, String newName)** | **Mail** | Renames a mail. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.renameMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "new name");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**moveMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **moveMail(String mailId, 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.moveMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a7986
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**copyMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **public void copyMail(String mailId, String dstId, String newName)** | **Mail** | Copies mail into a folder or record. |

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

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.copyMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a79861
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**extendedMailCopy**

Description:

| Method | Return values | Description |
|---|---|---|
| **extendedMailCopy(String mailId, String dstId, bool categories, bool keywords, bool propertyGroups, bool notes, bool wiki, bool security, String newName)** | **Mail** | Copies 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, 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 property groups parameter is true the original values of the metadata groups will be copied.

- When the notes parameter is true the original values of the notes will be copied.
- When the wiki parameter is true the original values of the wiki will be copied.
- When newName is set the mail name is renamed.
- When the security parameter is true the original value of the security will be copied.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.extendedMailCopy("fe51797c-be0b-4076-8f4b-b4fffe8fd2bc", "055e4384-7c70-4456-b32b-f5a55a
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMailChildren**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailChildren(String fldId)** | **List<Mail>** | Returns a list of all mails which their parent is fldId. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (Mail mail in ws.mail.getMailChildren("055e4384-7c70-4456-b32b-f5a55a79861f"))
            {
               System.Console.WriteLine(mail);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**isValidMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **isValidMail(String mailId)** | **Boolean** | Returns a boolean that indicates if the node is a mail or not. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            // Return true
            ws.mail.isValidMail("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getMailPath**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailPath(String mailId)** | **String** | Converts a mail UUID to folder path. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.mail.getMailPath("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createAttachment**

Description:

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

| | | |
|---|---|---|
| **createAttachment(String mailId, String docName, InputStream is)** | **Document** | Stores in the mail an attachment. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\logo.png", FileMode.Open);
            ws.mail.createAttachment("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "log.png", fs);
            fs.Dispose();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteAttachment**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteAttachment(String mailId, String docId)** | **void** | Deletes a mail attachment. |
| The value of the parameter **docId** parameter can be a valid document **UUID** or **path**. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.mail.deleteAttachment("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "e339f14b-4d3a-489c-91d3-05e4
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getAttachments**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (Document doc in ws.mail.getAttachments("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"))
            {
```

```
                System.Console.WriteLine(doc);
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**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)** | **Mail** | Sends a mail message with an attachement. |

The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored into the repository after be sent.

Other parameters:

- to are a list of mail accounts destination.
- subject is the mail subject.
- cc, bcc is a list of mail accounts destination ( optional )
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail ( optional ).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
```

```
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               List<String> to = new List<String>();
               to.Add("destination@mail.com");
               List<String> cc = new List<String>();
               List<String> bcc = new List<String>();
               List<String> docsId = new List<String>();
               List<String> replyTo= new List<String>();
               docsId.Add("7aa523e0-06cf-4733-b8c8-cc74d8b2716d");
               docsId.Add("f123a950-0329-4d62-8328-0ff500fd42db");
               ws.mail.sendMailWithAttachments(to, cc, bcc, replyTo, "some subject", "This mail its a test.", docsId, "055
            }
            catch (Exception e)
            {
               System.Console.WriteLine(e.ToString());
            }
         }
      }
   }
```

**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)** | **Mail** | Sends a mail message with an attachement. |

The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored into the repository after be sent.

Other parameters:

- to are a list of mail accounts destination.
- subject is the mail subject.
- cc, bcc is a list of mail accounts destination ( optional )
- docsId is a list of valid document UUID already into OpenKM that will be sent as an attachment into mail ( optional ).

Example:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            string from = "test@none.com";
            List<String> to = new List<String>();
            to.Add("destination@mail.com");
            List<String> cc = new List<String>();
            List<String> bcc = new List<String>();
            List<String> docsId = new List<String>();
            List<String> replyTo= new List<String>();
            docsId.Add("7aa523e0-06cf-4733-b8c8-cc74d8b2716d");
            docsId.Add("f123a950-0329-4d62-8328-0ff500fd42db");
            ws.mail.sendMailWithAttachments(from, to, cc, bcc, replyTo, "some subject", "This mail its a test.", docsId
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**importEml**

Description:

| Method | Return values | Description |
|---|---|---|
| **importEml(String dstId, String title, FileStream fs)** | **Mail** | Import a mail in EML format. |
| The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored in the repository after is sent. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream filestream = new FileStream("C:\\Documents\\test.eml", FileMode.Open);
            Mail mail = ws.mail.importEml("f123a950-0329-4d62-8328-0ff500fd42db", "Test", filestream);
            System.Console.WriteLine(mail);
            filestream.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**importMsg**

Description:

| Method | Return values | Description |
|---|---|---|
| **importMsg(String dstId, String title, FileStream fs)** | **Mail** | Import a mail in MSG format. |
| The values of the dstId parameter should be a folder or record UUID or path. The dstId parameter indicates where the mail will be stored in the repository after is sent. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream filestream = new FileStream("C:\\Documents\\test.msg", FileMode.Open);
                Mail mail = ws.mail.importMsg("f123a950-0329-4d62-8328-0ff500fd42db", "Test", filestream);
                System.Console.WriteLine(mail);
                filestream.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setMailTitle**

Description:

| Method | Return values | Description |
|---|---|---|
| **setMailTitle(String mailId, String title)** | **Mail** | Sets the mail title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.mail.setMailTitle("68ed7a93-005c-4ec6-ac01-bb151573c775", "new title");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**sendMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **sendMail(List<String> to, String subject, String body)** | **Void** | Sends a mail. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<String> to = new List<string>();
            to.Add("some@mail.com");
            ws.mail.sendMail(to, "Testing sending mail from OpenKM", "Test message body.");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**sendMail**

Description:

| Method | Return values | Description |
|---|---|---|
| **sendMail(String from, List<String> to, String subject, String body)** | **Void** | Sends a mail. |

Example:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            string from = "test@mome.com"
            List<String> to = new List<string>();
            to.Add("some@mail.com");
            ws.mail.sendMail(from, to, "Testing sending mail from OpenKM", "Test message body.");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setMailDescription**

Description:

| Method | Return values | Description |
|---|---|---|
| **setMailDescription(String mailId, String description)** | **void** | Sets the mail title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
```

```
            ws.login(user, password);
            ws.mail.setMailDescription("68ed7a93-005c-4ec6-ac01-bb151573c775", "Any description");
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getMailContent**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailContent(String mailId)** | **Stream** | Sets the mail title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
             ws.login(user, password);
             Stream s = ws.mail.getMailContent("68ed7a93-005c-4ec6-ac01-bb151573c775");
             FileStream mailFile = new FileStream("C:\\mailtest.msg", FileMode.OpenOrCreate);
             s.CopyTo(mailFile);
             s.Close();
             mailFile.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMailThumbnail**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailThumbnail(String mailId, String type)** | **Stream** | 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Stream s = ws.mail.getMailThumbnail("68ed7a93-005c-4ec6-ac01-bb151573c775", ThumbnailType.THUM
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createWizardMail**

Description:

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

| | | |
|---|---|---|
| **createWizardMail(String uuid, String title, FileStream fileStream, String type)** | **WizardNode** | Create a mail with wizard. |

The Wizard node contains the steps what might be followed by the wizard:

- Metadata groups
- Keywords
- Categories

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Folder fld = ws.createFolder("f123a950-0329-4d62-8328-0ff500fd42db", "Test");
            FileStream filestream = new FileStream("C:\\testing.msg", FileMode.Open, FileAccess.Read);
            WizardNode wn = ws.mail.createWizardMail(fld.uuid, "Any title", filestream, Mail.ORIGIN_MSG);
            filestream.Close();
            System.Console.WriteLine(wn.toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMailAccounts**

Description:

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

| | | |
|---|---|---|
| **getMailAccounts()** | **List<MailAccount>** | Retrieves a list of user email accounts. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (MailAccount mail in ws.mail.getMailAccounts())
            {
               System.Console.WriteLine(mail.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMailMessages**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailMessages(long accountId, long start)** | **MailServerMessages** | Retrieves a list of messages in the email server for an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            long accoundId = 1;
            long start = 1;
            MailServerMessages msm = ws.mail.getMailMessages(accoundId, start);
            System.Console.WriteLine(msm.toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**addMailAccount**

Description:

| Method | Return values | Description |
|---|---|---|
| **addMailAccount(MailAccount mailAccount)** | **void** | Add an email account. |

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

When you do **not set the user**, the account will be **added by default to the logged user**.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               MailAccount mailAcount = new MailAccount();
               mailAcount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
               mailAcount.mailUser = "test@none.com";
               mailAcount.mailPassword = "123456";
               mailAcount.mailHost = "imap.gmail.com";
               mailAcount.mailFolder = "OpenKM";
               mailAcount.active = true;
               mailAcount.recursive = true;
               ws.mail.addMailAccount(mailAcount);
            }
            catch (Exception e)
            {
               System.Console.WriteLine(e.ToString());
            }
         }
      }
   }
```

**updateMailAccount**

Description:

| Method | Return values | Description |
|---|---|---|
| **updateMailAccount(MailAccount mailAccount)** | **void** | Update the main configuration data of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            MailAccount mailAcount = new MailAccount();
```

```
            mailAcount.id = 1;// Valid mail account id;
            mailAcount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
            mailAcount.mailUser = "test@none.com";
            mailAcount.mailPassword = "123456";
            mailAcount.mailHost = "imap.gmail.com";
            mailAcount.mailFolder = "OpenKM";
            mailAcount.active = false;
            mailAcount.mailMarkDeleted = false;
            mailAcount.recursive = false;
            ws.mail.updateMailAccount(mailAcount);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**testMailAccount**

Description:

| Method | Return values | Description |
|---|---|---|
| **testMailAccount(MailAccount mailAccount)** | **void** | Verify connectivity of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            MailAccount mailAcount = new MailAccount();
            mailAcount.mailProtocol = MailAccount.PROTOCOL_IMAPS;
            mailAcount.mailUser = "test@none.com";
            mailAcount.mailPassword = "123456";
            mailAcount.mailHost = "imap.gmail.com";
            mailAcount.mailFolder = "OpenKM";
            mailAcount.active = true;
            mailAcount.recursive = true;

            // Test mail account
```

```
            ws.mail.testMailAccount(mailAcount);
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**deleteMailAccount**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteMailAccount(long mailAccountId)** | **void** | Delete an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            long mailId = 1; //Valid mail id
            ws.mail.deleteMailAccount(mailId);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**importMailMessages**

Description:

| Return |
|---|

| Method | values | Description |
|---|---|---|
| **importMailMessages(long mailAccountId, List<long> messageIds)** | **void** | Import messages of aa specific email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<long> messageIds = new List<long>();
            messageIds.Add(1);
            long mailAccountId = 1; // Valid mail id
            ws.mail.importMailMessages(mailAccountId, messageIds);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createMailFilter**

Description:

| Method | Return values | Description |
|---|---|---|
| **createMailFilter(long mailAccountId, MailFilter mailFilter)** | **void** | Add a filter of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Folder fld = ws.createFolder("f123a950-0329-4d62-8328-0ff500fd42db", "FilterFolder");
            // Valid mail account id
            long mailAccountId = 1;
            MailFilter filter = new MailFilter();
            filter.order = 0;
            filter.grouping = false;
            filter.exclusive = true;
            filter.active = false;
            filter.node = fld.uuid;
            ws.mail.createMailFilter(mailAccountId, filter);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**updateMailFilter**

Description:

| Method | Return values | Description |
|---|---|---|
| **updateMailFilter(MailFilter mailFilter)** | **void** | Update the filter of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                MailFilter filter = new MailFilter();
                filter.id = 1; // Valid filter id
                filter.grouping = true;
                filter.exclusive = false;
                filter.active = true;
                ws.mail.updateMailFilter(filter);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteMailFilter**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteMailFilter(long mailFilterId)** | **void** | Delete a filter of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                long filterId = 1; // Valid filter id
                ws.mail.deleteMailFilter(filterId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
```

```
        }
    }
}
```

**createMailRule**

Description:

| Method | Return values | Description |
|---|---|---|
| **createMailRule(long filterId, MailFilterRule rule)** | **void** | Create a rule of an email filter. |

Mail account filter parameters:

- **Field**: Sets the mail field that will be evaluated by the rule.
  Available options are: *From, To, Subject and Content*
  - MailFilterRule.FIELD_FROM
  - MailFilterRule.FIELD_TO
  - MailFilterRule.FIELD_SUBJECT
  - MailFilterRule.FIELD_CONTENT
  - MailFilterRule.FIELD_ATTACHMENT
- **Operation:** Set the operation that will be done for the evaluation process.
  Available options are: *Contains* and *Equal*
  - MailFilterRule.OPERATION_EQUALS
  - MailFilterRule.OPERATION_NOT_EQUALS
  - MailFilterRule.OPERATION_CONTAINS
  - MailFilterRule.OPERATION_ENDS_WITH
  - MailFilterRule.OPERATION_STARTS_WITH
- ***Value***: *Sets the value that will be used for the evaluation process.*
- ***Active:*** *Sets rule enabled or not.*

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            MailFilterRule rule = new MailFilterRule();
            rule.operation = MailFilterRule.OPERATION_CONTAINS;
            rule.value = "test";
            rule.field = MailFilterRule.FIELD_FROM;
            rule.active = false;
            long filterId = 1;// Valid filter id
            ws.mail.createMailRule(filterId, rule);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**updateMailRule**

Description:

| Method | Return values | Description |
|---|---|---|
| **updateMailRule(MailFilterRule rule)** | **void** | Update a rule of an email coount. |

Mail account filter parameters:

- **Field**: Sets the mail field that will be evaluated by the rule.

  Available options are:

  - MailFilterRule.FIELD_FROM
  - MailFilterRule.FIELD_TO
  - MailFilterRule.FIELD_SUBJECT
  - MailFilterRule.FIELD_CONTENT
  - MailFilterRule.FIELD_ATTACHMENT

- **Operation:** Set the operation that will be done for the evaluation process.

Available options are:

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

- ***Value**: Sets the value that will be used for the evaluation process.*
- ***Active:** Sets rule enabled or not.*

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<MailFilterRule> rules = ws.getMailFilterRules(filter.id);
            if (rules != null && rules.Count > 0)
            {
               // Update rule
               MailFilterRule rule = new MailFilterRule();
               rule.id = rules[0].id;
               rule.operation = MailFilterRule.OPERATION_EQUALS;
               rule.value = "Any";
               rule.field = MailFilterRule.FIELD_SUBJECT;
               rule.active = true;
               ws.mail.updateMailRule(rule);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteMailRule**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteMailRule(long ruleId)** | **void** | Delete a rule of an email account. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            long ruleId = 1; // Valid rule id
            ws.mail.deleteMailRule(ruleId);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMailFilterRules**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailFilterRules(long filterId)** | **List<MailFilterRule>** | Retrieve a list of all the associated rules of type filter. |

Example:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            long filterId = 1; // Valid filter id
            foreach (MailFilterRule mfrule in ws.mail.getMailFilterRules(filterId))
            {
               System.Console.WriteLine(mfrule.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getPdf**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPdf(String uuid)** | **Stream** | Returns a PDF of the email. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
```

```
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Mail mail = ws.mail.getMailProperties("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43");
                Stream stream = ws.mail.getPdf(mail.uuid);
                FileStream mailFile = new FileStream("D:\\" + mail.subject + ".pdf", FileMode.OpenOrCreate, FileAccess.R
                stream.CopyTo(mailFile);
                stream.Close();
                mailFile.Close();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Node samples

## Basics

Example of uuid:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

```
ws.node.getNodeByUuid("29a22996-0e3b-421e-8759-c24ea41c1ebb")
```

## Methods

**getVersionHistory**

Description:

| Method | Return values | Description |
|---|---|---|
| **getVersionHistory(String uuid)** | **List<Version>** | Returns a list of the version history of a document. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<Version> versions = ws.node.getVersionHistory("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**restoreVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **restoreVersion(String uuid, String versionName)** | **void** | Restore a document to a specific version. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.node.restoreVersion("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7","1.0");
```

```
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteVersion(String uuid, String versionName)** | **void** | Delete a specific version. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.deleteVersion("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7","1.0");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**purgeVersionHistory**

Description:

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

| | | |
|---|---|---|
| **purgeVersionHistory(String uuid)** | **void** | Purge version history of a document. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.node.purgeVersionHistory("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**subscribe**

Description:

| Method | Return values | Description |
|---|---|---|
| **subscribe(String uuid)** | **void** | Adds a subscription to a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.subscribe("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**unsubscribe**

Description:

| Method | Return values | Description |
|---|---|---|
| **unsubscribe(String uuid)** | **void** | Delete a subscription to a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.node.unsubscribe("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**importZip**

Description:

| Method | Return values | Description |
|---|---|---|
| **importZip(String uuid, FileStream content)** | **void** | Import a zip file. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            String folderId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7";
            FileStream zipFile = new FileStream("D:\\Test.zip", FileMode.Open, FileAccess.Read);
            ws.node.importZip(folderId, zipFile);
            zipFile.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**exportZip**

Description:

| Method | Return values | Description |
|---|---|---|
| **exportZip(List<String> uuids, bool withPath, bool background)** | **Stream** | Export as a zip file. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<string> uuids = new List<string>();
            uuids.Add("70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7");
            uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
            // Get strem exportZip
            Stream s = ws.node.exportZip(uuids, true, true);
            // Save as zip file
            FileStream zipFile = new FileStream("D:\\Testing\\Test.zip", FileMode.OpenOrCreate, FileAccess.ReadW
            s.CopyTo(zipFile);
            zipFile.Close();
            s.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**unZip**

Description:

| Method | Return values | Description |
|---|---|---|
| **unZip(String uuid, String dstId)** | **void** | Unzip file. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
```

```
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                String zipFileId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                String folderId = "70ec54af-76ad-4d02-b9c8-8c94c3b6ffc7";
                ws.node.unZip(zipFileId, folderId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getNodeByUuid**

Description:

| Method | Return values | Description |
|---|---|---|
| **getNodeByUuid(String uuid)** | **Node** | Get a node by uuid. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Node node = ws.node.getNodeByUuid("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
                System.Console.WriteLine(node.toString());
            }
            catch (Exception e)
            {
```

```
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getChildrenNodesPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes)** | **SimpleNodeBaseList** | Get children nodes paginated. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<int> filteredTypes = new List<int>();
                filteredTypes.Add(1);
                String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
                ChildNodeList nodeList = ws.node.getChildrenNodesPaginated(folderId, 0, 10, "", NodesPaginationInfo.O

            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getChildrenNodesPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **getChildrenNodesPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes, String pluginName)** | **SimpleNodeBaseList** | Get children nodes paginated. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<int> filteredTypes = new List<int>();
            filteredTypes.Add(1);
            String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
            ChildNodeList nodeList = ws.node.getChildrenNodesPaginated(folderId, 0, 10, "", NodesPaginationInfo.O

         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getChildrenNodesByCategoryPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes)** | **SimpleNodeBaseList** | Get children nodes by category paginated. |
| ✔ The parameter "limit" and "offset" allows you to retrieve just a portion of the results of a query. | | |

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<int> filteredTypes = new List<int>();
            filteredTypes.Add(1);
            String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
            ChildNodeList nodeList = ws.node.getChildrenNodesByCategoryPaginated(folderId, 0, 10, "", NodesPagir

         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getChildrenNodesByCategoryPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **getChildrenNodesByCategoryPaginated(String uuid, int offset, int limit, String filter, String orderByField, bool orderAsc, List<int> filteredTypes, String pluginName)** | **SimpleNodeBaseList** | Get children nodes by category paginated. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<int> filteredTypes = new List<int>();
            filteredTypes.Add(1);
            String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
            ChildNodeList nodeList = ws.node.getChildrenNodesByCategoryPaginated(folderId, 0, 10, "", NodesPagir

         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getBreadcrumb**

Description:

| Method | Return values | Description |
|---|---|---|
| **getBreadcrumb(String uuid)** | **List<BreadCrumbItem>** | Get bread crumb. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            String folderId = "82555dd1-bcc2-4e64-81cb-5f7c2b0d7801";
            List<BreadCrumbItem> list = ws.node.getBreadcrumb(folderId);
```

```
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getNodesFiltered**

Description:

| Method | Return values | Description |
|---|---|---|
| **getNodesFiltered(List<String> uuids)** | **List<Node>** | Return a node list. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<string> uuids = new List<string>();
                uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
                List<Node> nodes = ws.node.getNodesFiltered(uuids);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**evaluateDownloadZip**

Description:

| Method | Return values | Description |
|---|---|---|
| **evaluateDownloadZip(List<String> uuids)** | **ZipDownloadEvaluationResult** | Return a ZipDownloadEvaluationResult object. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<string> uuids = new List<string>();
            uuids.Add("82555dd1-bcc2-4e64-81cb-5f7c2b0d7801");
            ZipDownloadEvaluationResult zdeResult = ws.node.evaluateDownloadZip(uuids);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**restore**

Description:

| Method | Return values | Description |
|---|---|---|
| **restore(String uuid)** | **Node** | Restore a node |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Node node = ws.node.restore("0f6463f3-4d36-4091-b518-4fe7c353ee70");
            System.Console.WriteLine(node.toString());

         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**hasNodesLockedByOtherUser**

Description:

| Method | Return values | Description |
|---|---|---|
| **hasNodesLockedByOtherUser(String uuid)** | **bool** | If the node is blocked by other users |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
       static void Main(string[] args)
       {
          String host = "http://localhost:8080/openkm";
          String username = "okmAdmin";
          String password = "admin";
          OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

          try
          {
             ws.login(user, password);
             if(ws.node.hasNodesLockedByOtherUser("1ec49da9-1746-4875-ae32-9281d7303a62"))
```

```
            {
               System.Console.Writeline("Is blocked");
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setComment**

Description:

| Method | Return values | Description |
|---|---|---|
| **setComment(String uuid, String versionName, String comment)** | **void** | Sets the comment for a specific node version. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               ws.node.setComment("4b88cbe9-e73d-45fc-bac0-35e0d6e59e43", "1.5", "Update comment");
            }
            catch (Exception e)
            {
               System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Note samples

## Basics

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

Example of nodeId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**addNote**

Description:

| Method | Return values | Description |
|---|---|---|
| **addNote(String nodeId, String text)** | **Note** | Adds a note to a node and returns an object Note. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.note.addNote("f123a950-0329-4d62-8328-0ff500fd42db", "the note text");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getNote**

Description:

| Method | Return values | Description |
|---|---|---|
| **getNote(String noteId)** | **Note** | Retrieves the note. |

The noteId is a UUID.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Note> notes = ws.note.listNotes("f123a950-0329-4d62-8328-0ff500fd42db");
                if (notes.Count > 0)
                {
                    System.Console.WriteLine(ws.note.getNote(notes[0].path));
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteNote**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteNote(String noteId)** | **Note** | Deletes a note. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
```

```
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("f123a950-0329-4d62-8328-0ff500fd42db");
            if (notes.Count > 0)
            {
                ws.note.deleteNote(notes[0].path);
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**setNote**

Description:

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

The noteId is a UUID.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<Note> notes = ws.note.listNotes("f123a950-0329-4d62-8328-0ff500fd42db");
            if (notes.Count > 0)
            {
               ws.note.setNote(notes[0].path, "text modified");
            }
         }
         catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**listNotes**

Description:

| Method | Return values | Description |
|---|---|---|
| **listNotes(String nodeId)** | **List<Note>** | Retrieves a list of all notes of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<Note> notes = ws.note.listNotes("f123a950-0329-4d62-8328-0ff500fd42db");
                foreach (Note note in notes)
                {
                    System.Console.WriteLine(note);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getNotesHistory**

Description:

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

| | | |
|---|---|---|
| **getNotesHistory(String nodeId)** | **List<NoteHistory>** | Retrieves a list of all notes of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<NoteHistory> notes = ws.note.getNotesHistory("f123a950-0329-4d62-8328-0ff500fd42db");
                foreach (NoteHistory note in notes)
                {
                    System.Console.WriteLine(note);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Notification samples

## Basics

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**notify**

Description:

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

The parameter uuids are the UUID of the node ( document, folder, mail or record ).

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

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

The parameter mails are a set of email addresses - usually external mails - to be notified.

The parameter message is the content body of the mail.

When attachment value is true the node is attached into the mail.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<String> uuids = new List<string>();
            uuids.Add(doc.uuid);
            List<String> users = new List<string>();
            users.Add("jperez");
            List<String> roles = new List<string>();
            roles.Add("ROLE_USER");
            List<String> mails = new List<string>();
            mails.Add("test@none.com");
            String message = "Any message"
            ws.notification.notify(uuids, users, roles, mails, message, false);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

# Property samples

## Basics

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

Example of nodeId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**addCategory**

Description:

| Method | Return values | Description |
|---|---|---|
| **addCategory(String nodeId, String catId)** | **void** | Sets a relation between a category and a node. |
| The value of the catId parameter should be a category folder UUID or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.property.addCategory("f123a950-0329-4d62-8328-0ff500fd42db", "055e4384-7c70-4456-b32b-f5a55a
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**removeCategory**

Description:

| Method | Return values | Description |
|---|---|---|
| **removeCategory(String nodeId, String catId)** | **void** | Removes a relation between a category and a node. |
| The value of the catId parameter should be a category folder UUID or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
```

```
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.removeCategory("f123a950-0329-4d62-8328-0ff500fd42db", "ac165cab-a62a-4b17-89fc-248(
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**addKeyword**

Description:

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

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.addKeyword("f123a950-0329-4d62-8328-0ff500fd42db", "test");
```

```
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**removeKeyword**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.property.removeKeyword("f123a950-0329-4d62-8328-0ff500fd42db", "test");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setSigned**

Description:

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

| | | |
|---|---|---|
| **setSigned(String nodeId, boolean signed)** | **void** | Marks a document assigned or unsigned binary data into the repository |

The parameter nodeId should be a document node.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.property.setSigned("f123a950-0329-4d62-8328-0ff500fd42db", true);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**isSigned**

Description:

| Method | Return values | Description |
|---|---|---|
| **isSigned(String uuid)** | **bool** | Returns a boolean that indicates if the document is signed or not. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Is the document signed: " + ws.property.isSigned("6330d2a0-529f-4c14-baa1
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# PropertyGroup samples

## Basics

From an older OpenKM version we named "**Metadata Groups**" as "**Property Groups**".

Although we understand this name does not help a lot to identify these methods with metadata ones, for historical reason, we continue maintaining the nomenclature.

For more information about Metadata.

On almost methods, you'll see the parameter named "**nodeId**". 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**"

The class com.openkm.sdk4j.util.ISO8601 should be used to set and parse metadata date fields. The metadata field of type date values are stored into application in ISO-8601 basic format.

To convert retrieved metadata field of type date to a valid date use:

```
DateTime date = ISO8601.parseBasic(metadataFieldValue);
```

To save date value into the metadata field of type date use:

```
DateTime date = DateTime.now; // Present date
String metadataFieldValue = ISO8601.formatBasic(date);
// metadataFieldValue can be saved into repository metadata field of type date
```

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

other API methods.

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

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

## Methods

**addPropertyGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **addPropertyGroup(String nodeId, String grpName, Dictionary<String, String> propertiesMap)** | **void** | Adds a metadata group to a node. |

The grpName should be a valid Metadata group name.

It is not mandatory to set in the propertiesMap parameter all fields values, is enough with the fields you wish to change its values.

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="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups
```

To add several values on a metadata field of type **multiple** and to add value on a metadata field of type **simple** like this:

```
<select label="Multiple" name="okp:consulting.multiple" type="multiple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>

<select label="Simple" name="okp:consulting.simple" type="simple">
 <option label="One" value="one"/>
 <option label="Two" value="two"/>
```

```
 <option label="Three" value="three" />
</select>
```

You should do:

```
properties.Add("okp:consulting.multiple", "[one,two]");
properties.Add("okp:consulting.simple", "[one]");
```

Where "**one**" and "**two**" are valid values, the character "**,**" is used as a separator and the options should go between "**[**" "**]**".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<string, string> props = new Dictionary<string, string>();
            DateTime date = DateTime.Now;
            props.Add("okp:consulting.name", "test");
            props.Add("okp:consulting.date", ISO8601.formatBasic(date));
            props.Add("okp:consulting.important", "true");
            props.Add("okp:consulting.comment", "test");
            ws.propertyGroup.addPropertyGroup("f0064cf3-4776-44c2-868a-f08697a6957e", "okg:consulting", props)
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**removePropertyGroup**

Description:

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

| | | |
|---|---|---|
| **removePropertyGroup(String nodeId, String grpName)** | **void** | Removes a metadata group of a node. |
| The grpName should be a valid Metadata group name. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.propertyGroup.removePropertyGroup("f123a950-0329-4d62-8328-0ff500fd42db", "okg:consulting");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getPropertyGroups**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroups(String nodeId)** | **List<PropertyGroup>** | Retrieves a list of metadata groups assigned to a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (PropertyGroup pGroup in ws.propertyGroup.getPropertyGroups("f123a950-0329-4d62-8328-0ff5
            {
          System.Console.WriteLine(pGroup.name);
       }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getAllPropertyGroups**

Description:

| Method | Return values | Description |
|---|---|---|
| **getAllPropertyGroups()** | **List<PropertyGroup>** | Retrieves a list of all metadata groups set into the application. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (PropertyGroup pGroup in ws.propertyGroup.getAllPropertyGroups())
            {
               System.Console.WriteLine(pGroup.name);
```

```
        }
          }
          catch (Exception e)
          {
             System.Console.WriteLine(e.ToString());
          }
       }
    }
}
```

**getPropertyGroupForm**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupForm(String grpName)** | **List<FormElement>** | Retrieves a list of all metadata group elements definition. |

The grpName should be a valid Metadata group name.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupForm("okg:consulting"))
            {
               System.Console.WriteLine(fElement);
            }
         }
         catch (Exception e)
         {
```

```
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getPropertyGroupForm**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupForm(String uuid, String grpName)** | **List<FormElement>** | Retrieves a list of all metadata group elements definition. |

The grpName should be a valid Metadata group name.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupForm("f123a950-0329-4d62-8328
            {
               System.Console.WriteLine(fElement);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
```

```
}
```

**getPropertyGroupFormDefinition**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupFormDefinition(String grpName)** | **List<FormElement>** | Retrieves a list of all metadata group elements definition. |

The grpName should be a valid Metadata group name.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupFormDefinition("okg:consulting"))
                {
                    System.Console.WriteLine(fElement);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getPropertyGroupFormDefinition**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupFormDefinition(String uuid, String grpName)** | **List<FormElement>** | Retrieves a list of all metadata group elements definition. |

The grpName should be a valid Metadata group name.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.bean.form;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (FormElement fElement in ws.propertyGroup.getPropertyGroupFormDefinition("afe29d6e-7769-4
            {
               System.Console.WriteLine(fElement);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setPropertyGroupProperties**

Description:

| Method |
|---|

**setPropertyGroupProperties(String nodeId, String grpName, Dictionary<String, String> properties)**

The grpName should be a valid Metadata group name.

Before changing metadata, you **should have the group added in the node** ( see addGroup method ) otherwise wil

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

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="Consulting" name="okg:consulting">
    <input label="Name" type="text" name="okp:consulting.name"/>
    <input label="Date" type="date" name="okp:consulting.date" />
    <checkbox label="Important" name="okp:consulting.important"/>
    <textarea label="Comment" name="okp:consulting.comment"/>
  </property-group>
</property-groups>
```

To add several values on a metadata field of type **multiple** and to add value on a metadata field of type **simple** like

```
<select label="Multiple" name="okp:consulting.multiple" type="multiple">
  <option label="One" value="one"/>
  <option label="Two" value="two"/>
  <option label="Three" value="three" />
</select>

<select label="Simple" name="okp:consulting.simple" type="simple">
 <option label="One" value="one"/>
 <option label="Two" value="two"/>
 <option label="Three" value="three" />
</select>
```

You should do:

```
properties.Add("okp:consulting.multiple", "[one,two]");
properties.Add("okp:consulting.simple", "[one]");
```

Where "**one**" and "**two**" are valid values, the character "**,**" is used as a separator and the options should go between

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<String, String> properties = new  Dictionary<String, String>();
            properties.Add("okp:consulting.name", "new name");

       //The date fields must be saved with basic ISO 8601 format
  properties.Add("okp:consulting.date", ISO8601.formatBasic(DateTime.Now));
       ws.propertyGroup.setPropertyGroupPropertiesSimple("f123a950-0329-4d62-8328-0ff500fd42db","okg:consulti
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**hasPropertyGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **hasPropertyGroup(String nodeId, String grpName)** | **Boolean** | Returns a boolean that indicates if the node has or not a metadata group. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
```

```
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Have metadata group:"+ ws.propertyGroup.hasPropertyGroup("f123a950-032
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getRegisteredPropertyGroupDefinition**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRegisteredPropertyGroupDefinition()** | **String** | Return the XML Metada groups definition. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.propertyGroup.getRegisteredPropertyGroupDefinition());
            }
```

```
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**registerPropertyGroupDefinition**

Description:

| Method | Return values | Description |
|---|---|---|
| **registerPropertyGroupDefinition(InputStream is, String pgName)** | **void** | Sets the XML Metadata groups definition into the repository.. |

The method can only be executed by superuser grants ( ROLE_ADMIN member user).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                FileStream fs = new FileStream("E:\\PropertyGroups.xml",FileMode.Open);
                ws.propertyGroup.registerPropertyGroupDefinition(fs,"okg:testing");
                fs.Dispose();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getPropertyGroupSuggestions**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupSuggestions(String nodeId, String grpName, String propName)** | **List<String>** | Retrieves a list of a 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
```

```
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach(String value in ws.propertyGroup.getPropertyGroupSuggestions("f123a950-0329-4d62-8328-0ff5(
                {
                    System.Console.WriteLine(value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getPropertyGroupProperties**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupProperties(String nodeId, String grpName)** | **Dictionary<String, String>** | Retrieves a dictionary- (key,value) pairs - with Metada group values of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new  Dictionary<String, String>();
                properties = ws.propertyGroup.getPropertyGroupProperties("f123a950-0329-4d62-8328-0ff500fd42db","ol

                foreach (KeyValuePair<String, String> pair in properties)
                {
```

```
                System.Console.WriteLine(pair.Key + "-> " + pair.Value);
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getPropertyGroupPropertiesByVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupPropertiesByVersion(String nodeId, String grpName,String versionName)** | **Dictionary<String, String>** | Retrieves a dictionary- (key,value) pairs - with Metada group values of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<String, String> properties = new  Dictionary<String, String>();
            properties = ws.propertyGroup.getPropertyGroupPropertiesByVersion("f123a950-0329-4d62-8328-0ff500

            foreach (KeyValuePair<String, String> pair in properties)
            {
                System.Console.WriteLine(pair.Key + "-> " + pair.Value);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
```

```
}
```

**getPropertyGroupsByVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroupsByVersion(String nodeId, String versionName)** | **List<PropertyGroup>** | Retrieves a list of metadata groups assigned to a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (PropertyGroup pGroup in ws.propertyGroup.getPropertyGroupsByVersion("f123a950-0329-4d62
                {
             System.Console.WriteLine(pGroup.name, "1.1");
        }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getPropertyGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPropertyGroup(String grpName)** | **PropertyGroup** | Get property group definition. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            PropertyGroup pg =  ws.propertyGroup.getPropertyGroup("okg:consulting");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            String grpName = "okg:consulting";
            String propertyName = "okp:consulting.suggestbox";
            String key = "es";
            String result = ws.propertyGroup.getSuggestBoxKeyValue(grpName, propertyName, key);
            Console.WriteLine("Value: " + result);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getSuggestBoxKeyValuesFiltered**

Description:

| Method | Return values | Description |
|---|---|---|
| **getSuggestBoxKeyValuesFiltered(String grpName, String propertyName, String filter)** | **Dictionary<String, String>** | Retrieves a dictionary - (key, value) pairs - with 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            String grpName = "okg:consulting";
            String propertyName = "okp:consulting.suggestbox";
            String filter = "Port";
            Dictionary<String, String> result = ws.propertyGroup.getSuggestBoxKeyValuesFiltered(grpName, property
            foreach(var item in result)
            {
               Console.WriteLine(item.key + ":" + item.value);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
```

```
        }
    }
}
```

**validateField**

| Method | Return values | |
|---|---|---|
| **validateField(String value, String className, List<String> uuids)** | **String** | Ret |

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

Example of the Form validator implementation

More information at Creating your own Form Validator plugin.

In this example it will not be allowed two equal comments in the metadata field named okp:tecnology.comment

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

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

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

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

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

@PluginImplementation
public class DuplicateDocumentNumberValidator extends BasePlugin implements 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                List<String> uuids = new List<string>();
                uuids.Add("28ece92f-9708-4d3f-8033-18dc98b9e7f5");
                uuids.Add("5484b622-7c5f-425a-9451-7611c0caf227");

                String value = "test";
                String className = "com.openkm.plugin.form.validator.DuplicateDocumentNumberValidator";
                String message = ws.propertyGroup.validateField(value, className, uuids);
                Console.WriteLine("Validate: " + message);
            }
            catch (Exception e)
            {
```

```
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Record samples

On most methods, you'll see the parameter named "**recId**" and "**fldId**". The value of these parameters can be a valid record and folder **UUID**.

Example of recId:

- Using UUID -> "**28bb0c8b-3701-4457-a81a-fef7bc56ff33**"

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**createRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **createRecord(String fldId, String name, String title, long nodeClass)** | **Record** | Creates a new record and returns as a result an object Record. |

Optionally it can be set a title variable, the other variables of the Record ( record ) will not take any effect on record creation.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.record.createRecord(fldId, "PKI-100200", "some title", 0);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getRecordProperties**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRecordProperties(String recId)** | **Record** | Returns the record properties. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.record.getRecordProperties("50b7a5b9-89d2-430e-bbc9-6a6e01662a71").
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteRecord(String recId)** | **void** | Deletes a record. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.deleteRecord("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**purgeRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **purgeRecord(String recId)** | **void** | Records are definitely removed from the repository. |

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.record.purgeRecord("50b7a5b9-89d2-430e-bbc9-6a6e01662a71");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**renameRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **renameRecord(String recId, String newName)** | **void** | Renames a record. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.record.renameRecord("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "PKI-100201");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**moveRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **moveRecord(String recId, String dstId)** | **void** | Moves a record into a folder or record. |
| The values of the dstId parameter should be a folder or record UUID or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               ws.record.moveRecord("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a
            }
            catch (Exception e)
            {
               System.Console.WriteLine(e.ToString());
            }
         }
      }
   }
```

**copyRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **copyRecord(String recId, String dstId, String newName)** | **Record** | Copies a record into a folder or record. |

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.record.copyRecord("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5a55a7
```

```
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**isValidRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **isValidRecord(String recId)** | **Boolean** | Returns a boolean that indicates if the node is a record or not. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine("Is a record:"+ ws.record.isValidRecord("50b7a5b9-89d2-430e-bbc9-6a6e0166
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getRecordChildren**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRecordChildren(String fldId)** | **List<Record>** | Returns a list of all records which their parent is fldId |

The parameter **fldId** can be a folder or a record node.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (Record rec in ws.record.getRecordChildren("055e4384-7c70-4456-b32b-f5a55a79861f"))
            {
               System.Console.WriteLine(rec);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setRecordTitle**

Description:

| Method | Return values | Description |
|---|---|---|
| **setRecordTitle(String recId, String title)** | **void** | Sets a record title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
```

```
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.setRecordTitle("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "some title");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getRecordPath**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRecordPath(String uuid)** | **String** | Converts a record UUID to record path. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.record.getRecordPath("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
```

```
}
```

**setRecordDescription**

Description:

| Method | Return values | Description |
|---|---|---|
| **setRecordDescription(String recId, String description)** | **void** | Set the title. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.record.setRecordDescription("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "some description");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createWizardRecord**

Description:

| Method | Return values | Description |
|---|---|---|
| **createWizardRecord(String uuid, String name, String title, long nodeClass)** | **WizardNode** | Create a record with wizard. |

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

- Add keyword
- Add Categories
- Add Metadata

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            WizardNode wnode = ws.record.createWizardRecord("f123a950-0329-4d62-8328-0ff500fd42db", "REC-0
            System.Console.WriteLine(wnode.toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**createRecordFromTemplate**

Description:

| Method | Return values | Description |
|---|---|---|
| **createRecordFromTemplate(String uuid, String dstPath, bool categories, bool keywords, bool notes, bool propertyGroups, bool security, Dictionary<String, String> properties)** | **Record** | Creates a new record from the template and returns an object Record. |

The **uuid** parameter is the template UUID of the record.

The **dstPath** can be a folder or a record path.

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

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

Additional:

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

Example:

```
using System; using System.Collections.Generic;
using System.Linq; using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
            try
            {
                ws.login(user, password);
                Dictionary<String, String> properties = new Dictionary<String, String>();
                properties.Add("okp:tpl.name", "Some name");
                DateTime date = DateTime.Now;
                // Value must be converted to String ISO 8601 compliant
                properties.Add("okp:tpl.bird_date", ISO8601.formatBasic(date));
                properties.Add("okp:tpl.language", "[ \"java\" ]");
                Record record = ws.record.createRecordFromTemplate("9fa9787e-d8b0-4ff7-905a-a89f0b228ec8", "/ok
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**extendedRecordCopy**

Description:

| Method | Return values | Description |
|---|---|---|
| **extendedRecordCopy(String uuid, String dstId, String newName, bool categories, bool keywords, bool propertyGroups, bool notes, bool security)** | **Record** | Copies a record with the associated data into some folder or record. |

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

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

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

Additional:

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

Example:

```
using System; using System.Collections.Generic;
using System.Linq; using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
            try
            {
                ws.login(user, password);
                ws.record.extendedRecordCopy("ada67d44-b081-4b23-bdc1-74181cafbc5d", "8599eab7-ae61-4628-8
                  "new name record", true, true, true, true, true, true);
            }
            catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**createMissingRecords**

Description:

| Method | Return values | Description |
|---|---|---|
| **createMissingRecords(String recPath)** | **String** | Create missing records. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.record.createMissingRecords("/okm:root/rec001/rec0011/rec00111");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Relation samples

## Basics

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

Example of nodeId:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**getRelationTypes**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRelationTypes(String type)** | **List<RelationType>** | Retrieves a list of all relations defined of a type. |
| Available types values: | | |

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

More information on Relation types.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (RelationType type in ws.relation.getRelationTypes(RelationType.PARENT_CHILD))
            {
               System.Console.WriteLine(type);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**add-relation**

Description:

| Method | Return values | Description |
|---|---|---|
| **add-relation(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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (RelationType type in ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL))
            {
               // looking for a relation named invoice
               if (type.title.Equals("invoice"))
               {
                  ws.relation.addRelation("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteRelation**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteRelation(long relationId)** | **void** | Deletes a relationship. |

Only when any node will not use the relation can it be deleted? Otherwise, you'll get an error.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (RelationType type in ws.relation.getRelationTypes(RelationType.BIDIRECTIONAL))
            {
               // looking for a relation named invoice
               if (type.title.Equals("invoice"))
               {
                  ws.relation.deleteRelation(type.id);
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getRelations**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRelations(String nodeId)** | **List<Relation>** | Retrieves a list of all relations of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (Relation relation in ws.relation.getRelations("055e4384-7c70-4456-b32b-f5a55a79861f"))
                {
                    System.Console.WriteLine(relation);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getRelationGroups**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRelationGroups(String nodeId)** | **List<RelationGroup>** | Retrieves a list of all related groups of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("055e4384-7c70-4456-b32b-f5a55a7986
                {
                    System.Console.WriteLine(rGroup.toString());
                }
            }
            catch (Exception e)
```

```
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**addRelationGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **addRelationGroup(String nodeId, String groupName, long type)** | **void** | Adds a relation group at a node. |

On a relation group only makes sense to apply a relation type of RelationType.MANY_TO_MANY.

More information on Relation types.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (RelationType type in ws.relation.getRelationTypes(RelationType.MANY_TO_MANY))
            {
               if (type.title.Equals("staple"))
               {
                  ws.relation.addRelationGroup("055e4384-7c70-4456-b32b-f5a55a79861f", "staple group", type.id);
               }
            }
         }
         catch (Exception e)
         {
```

```
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**addNodeToGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **addNodeToGroup(String nodeId, long groupId)** | **void** | Adds a node to an existing relation group. |

A relation group only has the sense to apply the type RelationType.MANY_TO_MANY.

More information on Relation types.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (RelationGroup rGroup in ws.relation.getRelationGroups("055e4384-7c70-4456-b32b-f5a55a7986
            {
               if (rGroup.name.Equals("staple group"))
               {
                  ws.relation.addNodeToGroup("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", rGroup.id);
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
```

```
}
```

**deleteRelationGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteRelationGroup(long groupId)** | **void** | Removes a node from a related group. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662
                {
                    if (rGroup.name.Equals("staple group"))
                    {
                        ws.relation.deleteRelationGroup(rGroup.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**findRelationGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **findRelationGroup(long groupId)** | **RelationGroup** | Finds a relation group by id. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            RelationGroup rg = new RelationGroup();
            foreach (RelationGroup rGroup in ws.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662a71"))
            {
               if (rGroup.name.Equals("staple group3"))
               {
                  rg = ws.relation.findRelationGroup(rGroup.id);
                  break;
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**setRelationGroupName**

Description:

| Method | Return values | Description |
|---|---|---|
| **setRelationGroupName(long groupId, String groupName)** | **void** | Changes the relation group name. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
```

```
namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            RelationGroup rg = new RelationGroup();
            foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662
            {
               if (rGroup.name.Equals("staple group3"))
               {
                  ws.relation.setRelationGroupName(rGroup.id, "newGroup");
                  rg = ws.relation.findRelationGroup(rGroup.id);
                  break;
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getAllRelationGroups**

Description:

| Method | Return values | Description |
|---|---|---|
| **getAllRelationGroups(int relationTypeId, String filter, int offset, int limit)** | **RelationGroupResultSet** | Retrieves a list of all related groups. |

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            int relationTypeId = 1;
            String filter = "";
            RelationGroupResultSet resultSet = ws.relation.getAllRelationGroups(relationTypeId, filter, 0, 10);
            foreach (RelationGroup relationGroup in resultSet.results)
            {
               Console.WriteLine(relationGroup.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**deleteRelationGroupItem**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteRelationGroupItem(String nodeId, long groupId)** | **void** | Delete a node relation of a group. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (RelationGroup rGroup in ws.relation.getRelationGroups("50b7a5b9-89d2-430e-bbc9-6a6e01662
                {
                    if (rGroup.name.Equals("staple group3"))
                    {
                        ws.relation.deleteRelationGroupItem("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", rGroup.id);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Report samples

## Basics

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

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

**getReports**

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<Report> reports = ws.report.getReports(true);
            foreach (Report rep in reports)
            {
               System.Console.WriteLine(rep.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getReport**

Description:

| Method | Return values | Description |
|---|---|---|
| **getReport(long rpId)** | **Report** | Returns reports. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
              ws.login(user, password);
              List<Report> reports = ws.report.getReports(true);
              Report report = new Report();
              foreach (var rep in reports)
              {
                 if (rep.fileName.Equals("DocumentCheckout.rep"))
                 {
                   // Get report
                   report = ws.report.getReport(rep.id);
                   break;
                 }
              }
              System.Console.WriteLine(report.toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeReport**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeReport(long rpId, Dictionary<String, String> params, String format, String uuid)** | **Stream** | Return a document result of executing a report. |

Available formats:

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

- Report.FORMAT_TEXT

The parameter **uuid** is the UUID of a node ( this parameter is optional, usually has sense with reports executed from user interface where the results of the reports depend on the node selected ).

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
               ws.login(user, password);
               List<Report> reports = ws.report.getReports(true);
               Report report = new Report();
               foreach (var rep in reports)
               {
                  if (rep.fileName.Equals("DocumentCheckout.rep"))
                  {
                     // Get report
                     report = ws.report.getReport(rep.id);
                     break;
                  }
               }

               Dictionary<String, String> param = new   Dictionary<String, String>();
               param.Add("from_date", "2016-01-01");
               param.Add("to_date", "2016-12-23");
               String uuid = "";
               Stream stream = ws.report.executeReport(report.id, Report.FORMAT_PDF, param, uuid);

               BeanHelper beanHelper = new BeanHelper();
               Byte[] data = beanHelper.ReadToEnd(stream);
               FileStream fileStream = new FileStream(@"C:\Desktop\out.pdf", FileMode.OpenOrCreate, FileAccess.F
               foreach (byte b in data)
               {
                  fileStream.WriteByte(b);
               }
               fileStream.Close();
         } catch (Exception e) {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**generateDownloadReportToken**

Description:

| Method | Return values | Description |
|---|---|---|
| **generateDownloadReportToken(long rpId)** | **String** | Return the token for downloading the report. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            long repId = 1;
            String dwt = ws.report.generateDownloadReportToken(repId);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

# Repository samples

## Basics

### Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

### getRootFolder

Description:

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getRootFolder());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getTrashFolder**

Description:

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

The returned folder will be the user trash folder.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            System.Console.WriteLine(ws.repository.getTrashFolder());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**getTrashFolderBase**

Description:

| Method | Return values | Description |
|---|---|---|
| **getTrashFolderBase()** | **Folder** | Returns the object Folder of node "/okm:trash" |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getTrashFolderBase());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getTemplatesFolder**

Description:

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getTemplatesFolder());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getPersonalFolder**

Description:

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

The returned folder will be the user personal folder.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getPersonalFolder());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getPersonalFolderBase**

Description:

| Method | Return values | Description |
|---|---|---|
| **getPersonalFolderBase()** | **Folder** | Return the object Folder of node "/okm:personal" |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getPersonalFolderBase());
         }
         catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getMailFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailFolder()** | **Folder** | Returns the object Folder of node "/okm:mail/{userId}" |

The returned folder will be the user mail folder.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getMailFolder());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getMailFolderBase**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMailFolderBase()** | **Folder** | Returns the object Folder of node "/okm:mail" |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getMailFolderBase());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getCategoriesFolder**

Description:

| Method | Return values | Description |
|---|---|---|
| **getCategoriesFolder()** | **Folder** | Returns the object Folder of node "/okm:categories" |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
```

```
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getCategoriesFolder());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**purgeTrash**

Description:

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

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            try
            {
                ws.login(user, password);
                ws.repository.purgeTrash();
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getUpdateMessage());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
```

```
    }
}
```

**getRepositoryUuid**

Description:

| Method | Return values | Description |
|---|---|---|
| **getRepositoryUuid()** | **String** | Retrieves installation unique identifier. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getRepositoryUuid());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**hasNode**

Description:

| Method | Return values | Description |
|---|---|---|
| **hasNode(String nodeId)** | **Boolean** | Returns a node that indicates if a node exists or not. |
| The value of the parameter nodeId can be a valid **UUID** or **path**. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine("Exists node:" + ws.repository.hasNode("064ff51a-b815-4f48-a096-b4946876
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getNodePath**

Description:

| Method | Return values | Description |
|---|---|---|
| **getNodePath(String uuid)** | **String** | Converts a node UUID to the path. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
```

```
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getNodePath("e339f14b-4d3a-489c-91d3-05e4575709d2"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getNodeUuid**

Description:

| Method | Return values | Description |
|---|---|---|
| **getNodeUuid(String path)** | **String** | Convert node path to UUID. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                System.Console.WriteLine(ws.repository.getNodeUuid("/okm:root/tmp"));
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getAppVersion**

Description:

| Method | Return values | Description |
|---|---|---|
| **getAppVersion()** | **AppVersion** | Return information about OpenKM version. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getAppVersion().toString());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**copyAttributes**

Description:

| Method | Return values | Description |
|---|---|---|
| **copyAttributes(String nodeId, String dstId, boolean categories, boolean keywords, boolean propertyGroups, boolean notes, boolean wiki)** | **void** | Copy attributes from a node to other. |

The values of the dstId parameter should be a node UUID or path.

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.repository.copyAttributes("50b7a5b9-89d2-430e-bbc9-6a6e01662a71", "055e4384-7c70-4456-b32b-f5
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeScript**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeScript(FileStream fs)** | **ScriptExecutionResult** | Execute an script. |

The local script - test.bsh - used in the sample below:

```
import com.openkm.api.OKMFolder;
import com.openkm.bean.Folder;
import com.openkm.util.ContextWrapper;
```

```
try {
      OKMFolder okmFolder = ContextWrapper.getContext().getBean(OKMFolder.class);
      for (Folder fld : okmFolder.getChildren(null, "/okm:root")) {
        print(fld.getPath() + "\n");
      }
} catch (Exception e) {
 e.printStackTrace();
}

// Some value can also be returned as string
return "test result";
```

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\test.bsh", FileMode.Open);
            ScriptExecutionResult result = ws.repository.executeScript(fs);
            System.Console.WriteLine(result.result);
            System.Console.WriteLine(result.stdout);

            if (!result.stderr.Equals(""))
            {
               System.Console.WriteLine("Error happened");
               System.Console.WriteLine(result.stderr);
            }

            fs.Dispose();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeScript**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeScript(String script)** | **ScriptExecutionResult** | Executes an script. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(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);
            Console.WriteLine(result.result);
            Console.WriteLine(result.stdout);
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeSqlQuery**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeSqlQuery(FileStream fs)** | **SqlQueryResults** | Executes SQL sentences. |

The test.sql script used in the sample below:

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

The SQL script can only contain a single SQL sentence.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream("E:\\test.sql", FileMode.Open);
            SqlQueryResults result = ws.repository.executeSqlQuery(fs);
            fs.Dispose();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeSqlQuery**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeSqlQuery(String sql)** | **SqlQueryResults** | Executes SQL sentences. |

The SQL script can only contains a single SQL sentence.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using System.IO;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            SqlQueryResults result = ws.repository.executeSqlQuery("SELECT NBS_UUID, NBS_CONTEXT, NBS_N
            foreach (SqlQueryResultColumns columns in result.sqlQueryResults)
            {
               Console.WriteLine("UUID:" + columns.sqlQueryResultColumns[0]);
               Console.WriteLine("Context:" + columns.sqlQueryResultColumns[1]);
               Console.WriteLine("Name:" + columns.sqlQueryResultColumns[2]);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**executeHqlQuery**

Description:

| Method | Return values | Description |
|---|---|---|
| **executeHqlQuery(FileStream fs)** | **HqlQueryResults** | Execute HQL sentences. |

The test.sql script used in the sample below:

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

The HQL script can only contain a single HQL sentence.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            FileStream fs = new FileStream(@"C:\Desktop\test.sql", FileMode.Open);
            HqlQueryResults result = ws.repository.executeHqlQuery(fs);
            fs.Dispose();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            HqlQueryResults result = ws.repository.executeHqlQuery("SELECT uuid, name from NodeBase where na
            foreach (HqlQueryResultColumns row in result.hqlQueryResults)
            {
               Console.WriteLine("uuid: " + row.hqlQueryResultColumns[0] + ", name: " + row.hqlQueryResultColumn
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getTranslations**

Description:

| Method | Return values | |
|---|---|---|
| **getTranslations(String lang, String module)** | **Dictionary<String, String>** | Retrieves |

The OpenKM translations tables can be used to retrieve actually OpenKM translations or create your own translatio

By default modules values are :

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

Example to add a new Translation module :

SQL values to be executed from Database query view:

```
DELETE FROM OKM_TRANSLATION WHERE TR_LANGUAGE='en-GB' and TR_MODULE='doc';
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
INSERT INTO OKM_TRANSLATION (TR_MODULE, TR_KEY, TR_TEXT, TR_LANGUAGE) VALUES
```

The code then should be:

```
Dictionary<String, String> translations = ws.repository.getTranslations("en-GB", "doc");
```

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            Dictionary<string,string>translations = ws.repository.getTranslations("en-GB", "frontend");
            foreach (KeyValuePair<string, string> kvp in translations)
            {
               Console.WriteLine("key:{0},with translation:{1}", kvp.Key, kvp.Value);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
```

```
    }
}
```

**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**", will be restricted for non Administrator users what parameters are accessible. That means any non Administrator use who will try accessing across the webservices to configuration parameters not set into the list of values of "**webservices.visible.properties**" will get an access denied exception.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Configuration configuration = ws.repository.getConfiguration("system.ocr");
                System.Console.WriteLine(configuration.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getChangeLog**

Description:

| Method | Return values | Description |
|---|---|---|
| **getChangeLog(String nodePath, DateTime modificationsFrom)** | **List<ChangeLogged>** | Return the list of changes in some path and subfolders. |

- The method is used by the desktop synchronization application for retrieving the changes.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<ChangeLogged> changeLoggedList = ws.repository.getChangeLog("/okm:root/test", DateTime.Now);
            foreach (var cl in changeLoggedList)
            {
               System.Console.WriteLine(cl.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getServerTime()**

Description:

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

The server time returned format is ISO8601

The method is used by desktop synchronization application for retrieving the changes

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            System.Console.WriteLine(ws.repository.getServerTime());
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getAvailableLocales**

Description:

| Method | Return values | Description |
|---|---|---|
| **getAvailableLocales(String locale)** | **Dictionary<String, String>** | Return the available languages. |

Example:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Dictionary<String, String> locales = ws.repository.getAvailableLocales("en-GB");
                foreach(KeyValuePair<String, String> local in locales)
                {
                    Console.WriteLine("Language:" + local.Key + "," + local.Value);
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

# Search samples

## Basics

Most methods use QueryParams here there're some tips about how to use them.

| Variables | Type | Allow wildcards | R |
|---|---|---|---|
| **domain** | **long** | No. | Available values:<br>• QueryParams.DOCUMENT<br>• QueryParams.FOLDER<br>• QueryParams.MAIL<br>• QueryParams.RECORD<br>By default the value is set to QueryParams.DOCUMENT.<br>For searching documents and folders use value:<br>(QueryParams.DOCUMENT \| QueryParams.FOLDER) |
| **author** | **String** | No. | The value must be a valid userId. |
| **name** | **String** | Yes. | |
| **title** | **String** | Yes. | |
| **keywords** | **List<String>** | Yes. | |
| **categories** | **List<String>** | No. | Values should be categories UUID, not use path value. |

| | | | |
|---|---|---|---|
| **content** | | Yes. | |
| **mimeType** | | No. | The value should be a valid and registered MIME type.<br><br>Only can be applied to documents node. |
| **language** | | No. | The value should be a valid language.<br><br>Only can be applied to documents node. |
| **folder** | | No. | When empty is used by default "/okm:root" node.<br><br>The value should be a valid UUID, not use a path value. |
| **folderRecursive** | **Boolean** | No. | It only makes sense to set this variable to true when the variable f |
| **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** | **Dictionary<String, String>** | Yes on almost. | On metadata field values like "date" cannot be applied wildcards.<br><br>The dictionary of the properties is composed of pairs:<br><br>('metadata_field_name','metada_field_value")<br><br>For example:<br><br>`Dictionary<String, String> properties = new Dictionary<S`<br>`properties.Add("okp:consulting.name", "name value")`<br><br>**Filtering by a range of dates:**<br><br>`DateTime date = DateTime.Now;// today`<br>`DateTime to = new DateTime(date.Year, date.Month, da`<br>`DateTime from = to.AddDays(-3);// three days before`<br>`Dictionary<String,String> properties = new Dictionary<S`<br>`properties.Add("okp:consulting.date", ISO8601.formatBa`<br><br>When filtering by a range of dates you mu will be ignored from OpenKM side.<br><br>**To filtering by a metadata field of type multiple**<br><br>`<select label="Multiple" name="okp:consulting.multiple" t`<br>`<option label="One" value="one"/>`<br>`<option label="Two" value="two"/>`<br>`<option label="Three" value="three" />`<br>`</select>` |

You should do:

```
properties.Add("okp:consulting.multiple", "one;two");
```

Where "**one**" and "**two**" are valid values and chara

The search operation is done only by AND logic.

Wildcard examples:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

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

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

## Methods

### find

Description:

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

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

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshaling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.domain = QueryParams.DOCUMENT;
            qParams.name = "test*.html";
            foreach (QueryResult qr in ws.search.find(qParams, null))
            {
               System.Console.WriteLine(qr);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
```

```
    }
}
```

**find**

Description:

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

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

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

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

```
            QueryParams _params = new QueryParams();
            _params.domain = QueryParams.DOCUMENT + QueryParams.FOLDER;
            _params.name = "test*";
            foreach (QueryResult qr in ws.search.find(_params, SearchSortField.LAST_MODIFIED, true, null))
            {
                Console.WriteLine(qr.toString());
            }
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**findPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **findPaginated(QueryParams queryParams, int offset, int limit, String propertiesPlugin)** | **ResultSet** | Returns a list of paginated results filtered by the values of the queryParams parameter. |

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.
- The parameter "propertiesPlugin" must be the canonical class name of the class which implements the NodeProperties interface.

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshaling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.domain = QueryParams.DOCUMENT;
            qParams.name = "test*.html";
            ResultSet rs = ws.search.findPaginated(qParams, 0, 10, null);
            System.Console.WriteLine("Total results:" + rs.total);

            foreach (QueryResult qr in rs.results)
            {
               System.Console.WriteLine(qr);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **findPaginated(QueryParams queryParams, String sortField, bool sortReverse, int offset, int limit, String propertiesPlugin)** | **ResultSet** | Returns a list of paginated results filtered by the values of the queryParams parameter. |

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

- The parameter "limit" is used to limit the number of results returned.

- The parameter "offset" says to skip that many results before the beginning to return results.

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

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.domain = QueryParams.DOCUMENT + QueryParams.FOLDER;
            qParams.name = "test*";
```

```
            ResultSet rs = ws.search.findPaginated(params, SearchSortField.LAST_MODIFIED, true, 20, 10, null);
            System.Console.WriteLine("Total results:" + rs.total);

            foreach (QueryResult qr in rs.results)
            {
               System.Console.WriteLine(qr);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findSimpleNodeBasePaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **findSimpleNodeBasePaginated(QueryParams queryParams, int offset, int limit)** | **SimpleNodeBaseResultSet** | Returns a list of SimpleNodeBase paginated results filtered by the values of the queryParams parameter. |

Because the results of the findSimpleNodeBasePaginated method are objects of type SimpleNodeBase, this method is about 60-70% faster than the other search methods ( these metrics should be taken as orientative values because depends on hardware and the specific scenario where application is running). The other search methods returns objects of type Node what comes with a full node data, the SimpleNodeBase object provide less data but in almost cases should be enought. Specially when the limit is a higher value you will appreciate the difference in the perfomance.

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            QueryParams qParams = new QueryParams();
            qParams.domain = QueryParams.DOCUMENT;
            qParams.name = "test*";
            SimpleNodeBaseResultSet snbrs = ws.search.findSimpleNodeBasePaginated(qp, 0, 10);
            System.Console.WriteLine("Total results:" + snbrs.total);

            foreach (SimpleNodeBase snb in snbrs.results)
            {
               System.Console.WriteLine(snb.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
```

```
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
               ws.login(user, password);
               QueryParams qParams = new QueryParams();
               qParams.domain = QueryParams.DOCUMENT;
               qParams.name = "test*";
               SimpleNodeBaseResultSet snbrs = ws.search.findSimpleNodeBasePaginated(params, SearchSortField.A
               System.Console.WriteLine("Total results:" + snbrs.total);

               foreach (SimpleNodeBase snb in snbrs.results)
               {
                  System.Console.WriteLine(snb.toString());
               }
            }
            catch (Exception e)
            {
               System.Console.WriteLine(e.ToString());
            }
         }
      }
   }
```

**getKeywordMap**

Description:

| Method | Return values | Description |
|---|---|---|
| **getKeywordMap(String[] filter)** | **Dictionary<String, int>** | Returns a map of keywords with its count value filtered by other keywords. |

Example:

- Doc1.txt has keywords "test", "one", "two".
- Doc2.txt has keywords "test", "one"
- Doc3.txt has keywords "test", "three".

The results filtering by "test" -> "one", "two", "three".

The results filtering by "one" -> "test", "two".

The results filtering by "two" -> "test", "one".

The results filtering by "three" -> "test".

The results filtering by "one" and "two" -> "test".

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // All keywords without filtering
            System.Console.WriteLine("Without filtering");
            Dictionary<String, int> keywords = ws.search.getKeywordMap(new String[]{"test", "test2"});

            foreach (KeyValuePair<string,int> kvp in keywords)
            {
                System.Console.WriteLine(kvp.Key + " is used :" + kvp.Value);
            }

            // Keywords filtered
            System.Console.WriteLine("Filtering");
            keywords = ws.search.getKeywordMap(new String[]{"test"});

            foreach (KeyValuePair<string,int> kvp in keywords)
            {
               System.Console.WriteLine(kvp.Key + " is used :" + kvp.Value);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**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 or path. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8180/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (Document doc in ws.search.getCategorizedDocuments("50b7a5b9-89d2-430e-bbc9-6a6e01662a
            {
                System.Console.WriteLine(doc);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**saveSearch**

Description:

| Method | Return values | Description |
|---|---|---|
| **saveSearch(QueryParams params)** | **Long** | Saves search parameters. |
| The variable queryName of the parameter params should have to be initialized. | | |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
```

```
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                QueryParams qParams = new QueryParams();
                qParams.domain = QueryParams.DOCUMENT;
                qParams.name = "test*.html";

                foreach (QueryResult qr in ws.search.find(qParams))
                {
                    System.Console.WriteLine(qr);
                }

                // Save the search to be used later
                qParams.queryName = "sample search";
                ws.search.saveSearch(qParams);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**updateSearch**

Description:

| Method | Return values | Description |
|---|---|---|
| **updateSearch(QueryParams params)** | **void** | Updates a previously saved search parameters. |

Ir can only be updated as a saved search created by the same user who's executing the method.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryParams qParams in ws.search.getAllSearchs())
                {
                    if (qParams.queryName.Equals("sample search"))
                    {
                        // Change some value.
                        qParams.name = "admin*.html";
                        ws.search.updateSearch(qParams);
                    }
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getSearch**

Description:

| Method | Return values | Description |
|---|---|---|
| **getSearch(int qpId)** | **QueryParams** | Gets saved searches parameters. |

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
```

```
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int qpId = 1; // Some valid search id
                QueryParams qParams = ws.search.getSearch(qpId);
                System.Console.WriteLine(qParams);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**getAllSearchs**

Description:

| Method | Return values | Description |
|---|---|---|
| **getAllSearchs()** | **List<QueryParams>** | Retrieves a list of all saved search parameters. |

It can only retrieve the list of the saved searches created by the same user who's executing the method.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                foreach (QueryParams qParams in ws.search.getAllSearchs())
                {
                    System.Console.WriteLine(qParams.queryName);
                }
            }
            catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**deleteSearch**

Description:

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

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int qpId = 1; // Some valid search id
                ws.search.deleteSearch(qpId);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**findByQueryPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **findByQueryPaginated(String query, String sortField, boolean sortReverse, int offset, int limit, String propertiesPlugin)** | **ResultSet** | Returns a list of paginated results filtered by the values of the query parameter. |

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

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

More information about lucene sintaxis at Lucene query syntax.

Available sortField values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

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

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

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

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Example:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ResultSet rs = ws.search.findByQueryPaginated("text:grial AND name:t*.pdf", SearchSortField.NAME, tru
            Console.WriteLine("Total results:" + rs.total.ToString());
            foreach (QueryResult qr in rs.results)
            {
               Console.WriteLine(qr.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            SimpleNodeBaseResultSet snbr = ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND nar

            foreach (SimpleNodeBase nodeBase in snbr.results)
            {
               if(nodeBase != null)
               {
                  System.Console.WriteLine(nodeBase.toString());
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findSimpleNodeBaseByQueryPaginated**

Description:

| Method | Return values | Description |
|---|---|---|
| **findSimpleNodeBaseByQueryPaginated(String query, String sortField, bool 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:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            SimpleNodeBaseResultSet snbr= ws.search.findSimpleNodeBaseByQueryPaginated("text:grial AND nam

            foreach (SimpleNodeBase nodeBase in snbr.results)
            {
               if(nodeBase != null)
               {
                  System.Console.WriteLine(nodeBase.toString());
               }
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findByQuery**

Description:

| Method | Return values | Description |
|---|---|---|
| **findByQuery(String query, 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 - marshaling and unmarshalling process - while you are only interesting on using only a few Objects variables. If it's your case you can use NodeProperties classes for retrieving the Object variables what you really need.

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.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (QueryResult qr in ws.search.findByQuery("keyword:test AND name:t*.pdf", null))
            {
               System.Console.WriteLine(qr);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findByQuery**

Description:

| Method | Return values | Description |
|---|---|---|
| **findByQuery(String query, String sortField, bool 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (QueryResult qr in ws.search.findByQuery("keyword: test", SearchSortField.NAME, true, null))
            {
               System.Console.WriteLine(qr);
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findWithMetadata**

Description:

| Method | Return values | Description |
|---|---|---|
| **findWithMetadata(QueryParams _params, 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // QueryParams
            QueryParams qp = new QueryParams();
            qp.folderRecursive = true;
            qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryPara
            qp.name = "test*";

            // Properties
            Dictionary<string, string> properties = new Dictionary<string, string>();
            properties.Add("okp:consulting.name", "test");
            qp.properties = properties;

            // Groups
            List<string> groups = new List<string>();
            groups.Add("okg:consulting");

            List<QueryResult> queryResults = ws.search.findWithMetadata(qp, null, groups);
            foreach (QueryResult qResult in queryResults)
            {
               System.Console.WriteLine(qResult.toString());
            }
         }
         catch (Exception e)
```

```
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**findWithMetadata**

Description:

| Method | Return values | Description |
|---|---|---|
| **findWithMetadata(QueryParams queryParams, String sortField, bool 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:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                // QueryParams
                QueryParams qp = new QueryParams();
                qp.folderRecursive = true;
                qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryPara
                qp.name = "test*";

                // Properties
                Dictionary<string, string> properties = new Dictionary<string, string>();
                properties.Add("okp:consulting.name", "test");
                qp.properties = properties;

                // Groups
                List<string> groups = new List<string>();
                groups.Add("okg:consulting");

                List<QueryResult> queryResults = ws.search.findWithMetadata(qp, SearchSortField.NAME, true, null, gro
                foreach (QueryResult qr in queryResults)
                {
                    Console.WriteLine(qr.toString());
                }
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

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

- The parameter "limit" is used to limit the number of results returned.
- The parameter "offset" says to skip that many results before the beginning to return results.
- The parameter "propertiesPlugin" must be the canonical class name of the class which implements

the NodeProperties interface.

- The parameter "groups" must be valid metadata group names.

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshaling and unmarshalling process - while you are only interested in using only a few Object variables. If it's your case you can use NodeProperties classes for retrieving the Object variables that you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // QueryParams
            QueryParams qp = new QueryParams();
            qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryPara
            qp.name = "test*";

            // Properties
            Dictionary<string, string> properties = new Dictionary<string, string>();
            properties.Add("okp:consulting.name", "test");
            qp.properties = properties;

            // Groups
            List<string> groups = new List<string>();
            groups.Add("okg:consulting");
```

```
            ResultSet resulSet = ws.search.findWithMetadataPaginated(qp, 0, 10, null, groups);
            System.Console.WriteLine("Total results: " + rs.total.ToString());
        }
        catch (Exception e)
        {
            System.Console.WriteLine(e.ToString());
        }
    }
  }
}
```

**findWithMetadataPaginated**

Description:

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

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

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

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

The parameter "groups" must be valid metadata group names.

Available **sortField** values:

```
SearchSortField.NAME
SearchSortField.AUTHOR
SearchSortField.LAST_MODIFIED
```

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

- limit=10
- offset=0

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

- limit=10
- offset=10

Retrieving entire Objects ( Document, Folder, Record, Mail ) from REST can take a lot of time - marshalling and unmarshalling process - while you are only interesting on using only few Objects variables. If its your case you can use NodeProperties classes for retrieving the Object variables what you really need.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            // QueryParams
            QueryParams qp = new QueryParams();
            qp.domain = QueryParams.DOCUMENT | QueryParams.FOLDER | QueryParams.RECORD | QueryPara
            qp.name = "test*";

            // Properties
            Dictionary<string, string> properties = new Dictionary<string, string>();
            properties.Add("okp:consulting.name", "test");
            qp.properties = properties;

            // Groups
            List<string> groups = new List<string>();
            groups.Add("okg:consulting");

            ResultSet rs = ws.search.findWithMetadataPaginated(qp, SearchSortField.AUTHOR, true, 0, 10, null, grou
            Console.WriteLine("Total results: " + rs.total.ToString());
            foreach (QueryResult qr in rs.results)
            {
               Console.WriteLine(qr.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**getMimeTypes**

Description:

| Method | Return values | Description |
|---|---|---|
| **getMimeTypes()** | **List<MimeType>** | Retrieves a list of mimetypes. |

Only will be retrieved the list of the mimeTypes that may be used in the search.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            foreach (MimeType mimeType in ws.search.getMimeTypes())
            {
               System.Console.WriteLine(mimeType.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**csvExport**

Description:

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

| | | |
|---|---|---|
| **csvExport(String lang, QueryParams queryParams, bool compact)** | **InputStream** | Export as a csv a list of results filtered by the values of the queryParams parameter. |

The parameter **lang** must be ISO 691-1 compliant.

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

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            QueryParams _params = new QueryParams();
            _params.domain = QueryParams.DOCUMENT + QueryParams.FOLDER;
            _params.name = "test*";
            Stream stream = ws.search.csvExport("es-ES", _params, false);
            FileStream destFile = new FileStream("C:\\Testing\\export.csv", FileMode.OpenOrCreate);
            stream.CopyTo(destFile);
            destFile.Dispose();
            stream.Dispose();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

# UserConfig samples

## Basics

### Suggested code sample

First, you must create the webservice object:

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

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

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

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

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

```
ws.userConfig.getConfig();
```

## Methods

### getConfig

Description:

| Method | Return values | Description |
|---|---|---|
| **getConfig()** | **UserConfig** | Returns the user settings. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
```

```
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                UserConfig userCfg = ws.userConfig.getConfig();
                Console.WriteLine("User: " + userCfg.user);
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setHome**

Description:

| Method | Return values | Description |
|---|---|---|
| **setHome(String uuid)** | **void** | Set the user home node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8180/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.userConfig.setHome("53751cb7-c462-4320-ba46-4cc33e4667ae");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
```

```
    }
}
```

# Wizard samples

## Basics

Example of uuid:

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

**Suggested code sample**

First, you must create the webservice object:

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

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

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

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

At this point you can use all the Wizardmethods from "**wizard**" class as is shown below:

```
ws.wizard.findByUser();
```

## Methods

**findByUser**

Description:

| Method | Return values | Description |
|---|---|---|
| **findByUser()** | **List<WizardNode>** | Returns a list of nodes with a wizard. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
```

```
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            List<WizardNode> list = ws.wizard.findByUser();
            foreach (WizardNode wizardNode in list)
            {
               Console.WriteLine(wizardNode.toString());
            }
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**findByUserAndUuid**

Description:

| Method | Return values | Description |
|---|---|---|
| **findByUserAndUuid(String uuid)** | **WizardNode** | Get the wizard of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
```

```
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                WizardNode wn= ws.wizard.findByUserAndUuid("5458e52a-58a5-4b5d-ac89-e4e5837924a9");
                Console.WriteLine(wn.toString());
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**hasWizardByUserAndNode**

Description:

| Method | Return values | Description |
|---|---|---|
| **hasWizardByUserAndNode(String uuid)** | **bool** | Know if a node has a wizard. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                Console.WriteLine("Has a wizard active: " + ws.wizard.hasWizardByUserAndNode("5458e52a-58a5-4b5d
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
```

```
}
```

**deleteAll**

Description:

| Method | Return values | Description |
|---|---|---|
| **deleteAll()** | **void** | Remove all wizards of the user. |

Only wizards assigned to the user session will be removed.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.wizard.deleteAll();
            Console.WriteLine("Delete all");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**addShowWizardCategories**

Description:

| Method | Return values | Description |
|---|---|---|
| **addShowWizardCategories(String uuid, int order)** | **void** | Add show wizard categories in the wizard of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            int order = 0;
            ws.wizard.addShowWizardCategories("055b5206-35d0-4351-ba92-c304bbba7ddf", order);
            Console.WriteLine("addShowWizardCategories");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**removeShowWizardCategories**

Description:

| Method | Return values | Description |
|---|---|---|
| **removeShowWizardCategories(String uuid)** | **void** | Remove show wizard categories in the wizard of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.wizard.removeShowWizardCategories("055b5206-35d0-4351-ba92-c304bbba7ddf");
            Console.WriteLine("removeShowWizardCategories");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**addShowWizardKeywords**

Description:

| Method | Return values | Description |
|---|---|---|
| **addShowWizardKeywords(String uuid, int order)** | **void** | Add show wizard keywords in the wizard of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;
```

```
namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            int order = 0;
            ws.wizard.addShowWizardKeywords("055b5206-35d0-4351-ba92-c304bbba7ddf", order);
            Console.WriteLine("addShowWizardKeywords");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```

**removeShowWizardKeywords**

Description:

| Method | Return values | Description |
|---|---|---|
| **removeShowWizardKeywords(String uuid)** | **void** | Remove show wizard keywords in the wizard of a node. |

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);
```

```
            try
            {
                ws.login(user, password);
                ws.wizard.removeShowWizardKeywords("055b5206-35d0-4351-ba92-c304bbba7ddf");
                Console.WriteLine("removeShowWizardKeywords");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**addGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **addGroup(String uuid, String group, int order)** | **void** | Add metadata group in the wizard. |

The parameter **uuid** is the unique node identifier.

The parameter **group** is the group name.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                int order = 0;
                ws.wizard.addGroup("055b5206-35d0-4351-ba92-c304bbba7ddf", "okg:testing", order);
                Console.WriteLine("addGroup");
            }
            catch (Exception e)
```

```
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**removeGroup**

Description:

| Method | Return values | Description |
|---|---|---|
| **removeGroup(String uuid, String group)** | **void** | Remove metadata group in the wizard. |

> The parameter **uuid** is the unique node identifier.
>
> The parameter **group** is the group name.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
    public class Program
    {
        static void Main(string[] args)
        {
            String host = "http://localhost:8080/openkm";
            String username = "okmAdmin";
            String password = "admin";
            OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

            try
            {
                ws.login(user, password);
                ws.wizard.removeGroup("055b5206-35d0-4351-ba92-c304bbba7ddf", "okg:testing");
                Console.WriteLine("removeGroup");
            }
            catch (Exception e)
            {
                System.Console.WriteLine(e.ToString());
            }
        }
    }
}
```

**setAutostart**

Description:

| Method | Return values | Description |
|---|---|---|
| **setAutostart(String uuid)** | **void** | Set auto-start to the wizard. |

**Autostart** should be always **set as the last wizard option**.

Example:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using com.openkm.sdk4csharp;
using com.openkm.sdk4csharp.util;
using com.openkm.sdk4csharp.bean;
using com.openkm.sdk4csharp.impl;
using System.IO;

namespace OKMRest
{
   public class Program
   {
      static void Main(string[] args)
      {
         String host = "http://localhost:8080/openkm";
         String username = "okmAdmin";
         String password = "admin";
         OKMWebservice ws = OKMWebservicesFactory.newInstance(host);

         try
         {
            ws.login(user, password);
            ws.wizard.setAutostart("055b5206-35d0-4351-ba92-c304bbba7ddf");
            Console.WriteLine("setAutostart");
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
      }
   }
}
```