Web Services API Guide

CONTENTS

INTRODUCTION

Doccept - Integrated Document Management System includes a full Web Services API, which allows programmers to develop custom solutions based on the Doccept Document Management Software.

Using our Web Services API, you will be able create new applications and integrate your existing applications into Doccept. You will be able to perform functions such as retrieve documents, search for documents, and upload new documents with great ease. You will also be able to build advanced applications and connect in-house or third-party applications to Doccept.

The following are the benefits of Doccept Web Services API:

- Platform Independent for calling applications
- Calling applications can be remote
- Integration of existing applications to Doccept
- Allows for custom development
- Build GUI or Web based applications that talk to Doccept

WEB SERVICES API

SUMMARY

Doccept has a complete API exposed via Web services. This means you can call any of these API methods from any programming language that supports the SOAP protocol, like Java, PHP or Python among others. This feature makes it possible to create a custom client, or integrate with third-party applications like CRM or CMS.

The following is the list of Web Services API available with Doccept:

API	Description
DOCAuth	Methods related to authentication, granting and revoking privileges.
DOCDocument	Methods related to document management.
DOCFolder	Methods related to folder management.
DOCMail	Methods related to mail management.
DOCProperty	Methods related to document properties management.
DOCSearch	Methods related to repository search.

DOCAuth

DOCAuth API provides methods related to authentication, granting and revoking privileges.

These methods used to remotely access the Doccept repository as well as to get the authenticated Users and roles. API also provides methods to remotely grant permissions based on users or roles.

Summary of API methods:

1	String login(String user, String pass)
2	void logout(String token)
3	StringArray getUsers(String token)
4	StringArray getRoles(String token)
5	void grantUser(String token, String nodePath, String user, int permissions, boolean recursive)
6	void revokeUser(String token, String nodePath, String user, int permissions, boolean recursive)
7	BytePairArray getGrantedUsers(String token, String nodePath)
8	void grantRole(String token, String nodePath, String role, int permissions, boolean recursive)
9	void revokeRole(String token, String nodePath, String role, int permissions, boolean recursive)
10	BytePairArray getGrantedRoles(String token, String nodePath)

Methods:

String LOGIN(String user, String pass)

Login into the repository and gets an authorization token with user info for future API invocations

Parameters

- *user* - User name for login.
- *pass* - Password for login.

Returns

- A token with authorization session info for next API invocations.

Throws

- *UserAlreadyLoggerException* - If the user is already logged into the system.
- *AccessDeniedException* - If authorization fails.
- *RepositoryException* - If there is an error accessing to repository.

void logout(String token)

Log out from the repository. Invalidates the authorization token.

Parameters

- *token* - The session authorization token.

Returns

- none

Throws

- *AccessDeniedException* - If token is not valid.
- *RepositoryException* - If there is an error accessing to repository.

StringArray getUsers(String token)

Retrieves list of repository users.

Parameters

- *token* - The session authorization token.

Returns

- A collection of repository users.

Throws

- *RepositoryException* - If there is any error retrieving the users list.

StringArray getRoles(String token)

Retrieves list of repository roles.

Parameters

- *token* - The session authorization token.

Returns

- A collection of repository roles.

Throws

- *RepositoryException* - If there is any error retrieving the roles list.

void grantUser(String token, String nodePath, String user, int permissions, boolean recursive)

Add user permissions to a node.

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.
- *user* - User name which permissions are changed.
- *permissions* - A mask with the permissions to be added.
- *recursive* - If the nodePath indicates a folder, the permissions can be applied recursively.

Returns

- none

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid.
- *RepositoryException* - If there is any error accessing to the repository.

void revokeUser(String token, String nodePath, String user, int permissions, boolean recursive)

Revoke user permissions from a node.

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.
- *user* - User name which permissions are changed.
- *permissions* - A mask with the permissions to be removed.
- *recursive* - If the nodePath indicates a folder, the permissions can be revoked recursively.

Returns

- none

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid.
- *RepositoryException* - If there is any error accessing to the repository.

BytePairArray getGrantedUsers(String token, String nodePath)

Get user permissions from a node (document or folder).

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.

Returns

- A hashmap with pairs of user / permissions.

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid
- *RepositoryException* - If there is any error accessing to the repository.

void grantRole(String token, String nodePath, String role, int permissions, boolean recursive)

Grant role permissions for a node.

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.
- *role* - Role name for which permissions are changed.
- *permissions* - A mask with the permissions to be added.
- *recursive* - If the nodePath indicates a folder, the permissions can be applied recursively.

Returns

- none

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid
- *RepositoryException* - If there is any error accessing to the repository.

void revokeRole(String token, String nodePath, String role, int permissions, boolean recursive)

Revoke role permissions from a node.

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.
- *role* - Role name for which permissions are changed.
- *permissions* - A mask with the permissions to be removed.
- *recursive* - If the nodePath indicates a folder, the permissions can be applied recursively.

Returns

- none

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid.
- *RepositoryException* - If there is any error accessing to the repository.

BytePairArray getGrantedRoles(String token, String nodePath)

Get roles permissions from a node (document or folder).

Parameters

- *token* - The session authorization token.
- *nodePath* - The complete path to the node.

Returns

- A hashmap with pairs of role / permissions.

Throws

- *ItemNotFoundException* - If the node defined by nodePath do not exists.
- *AccessDeniedException* - If the token authorization information is not valid.
- *RepositoryException* - If there is any error accessing to the repository.

Usage:

Dot Net Client:

The following is sample code of making this API method call from a dot net client:

```
[System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://endpoint.ws.doccept.com/",
ResponseNamespace="http://endpoint.ws.doccept.com/", Use=System.Web.Services.Description.SoapBindingUse.Literal)]

    [return: System.Xml.Serialization.XmlElementAttribute("return")]

    public string login(string user, string password) {

        object[] results = this.Invoke("login", new object[] {

                user,

                password});

        return ((string)(results[0]));

    }
```

JAVA Client:

The following is sample code of making this API method call from a Java client:

```
@WebService(name = "DOCAuth", serviceName = "DOCAuth", targetNamespace = "http://ws.doccept.com")
public class AuthService {
        private static Logger log = LoggerFactory.getLogger(AuthService.class);

        @WebMethod
        public String login(@WebParam(name = "user") String user,
                        @WebParam(name = "password") String password) throws AccessDeniedException, RepositoryException,
                        DatabaseException {
                log.debug("login({}, {})", user, password);
```

```
        AuthModule am = ModuleManager.getAuthModule();
        String token = am.login(user, password);
        log.debug("login: {}", token);
        return token;
    }

    @WebMethod
    public void logout(@WebParam(name = "token") String token) throws RepositoryException,
            DatabaseException {
        log.debug("logout({})", token);
        AuthModule am = ModuleManager.getAuthModule();
        am.logout(token);
        log.debug("logout: void");
    }
```

RESTful Client:

The following are the RESTful command line calls for DOCAuth API.

To fetch the security info associated to a given node. First show granted users:

```
$ curl -u admin:admin -H "Accept: application/json" -X GET \

  http://localhost:8080/doccept/services/rest/auth/getGrantedUsers/3492d662-b58e-417c-85b6-930ad0c6c3cf
```

API calls to fetch the granted roles:

```
$ curl -u admin:admin -H "Accept: application/json" -X GET \

  http://localhost:8080/doccept/services/rest/auth/getGrantedRoles/3492d662-b58e-417c-85b6-930ad0c6c3cf
```

DOCDocument

DOCDocument API provides methods related to document management such as creating a document, checking out, checking in, renaming, versioning etc.,

Summary of API methods:

1	Document create(String token, Document doc, byte[] content)
2	Document createSimple(String token, String docPath, byte[] content)
3	void delete(String token, String docId)
4	void lock(String token, String docPath)
5	void unlock(String token, String docPath)
6	Document rename(String token, String docPath, String newName)
7	void move(String token, String docPath, String fldPath)
8	Document getProperties(String token, String docPath)
9	void setProperties(String token, Document doc)
10	void setContent(String token, String docPath, byte[] content)
11	byte[] getContent(String token, String docPath, boolean checkout)
12	byte[] getContentByVersion(String token, String docPath, String versionId)
13	void checkout(String token, String docPath)
14	void cancelCheckout(String token, String docPath)
15	Version checkin(String token, String docPath, String comment)
16	VersionArray getVersionHistory(String token, String docPath)
17	void restoreVersion(String token, String docPath, String versionId)
18	DocumentArray getChilds(String token, String fldId)
19	boolean isValid(String token, String docPath)
20	String getPath(String token, String uuid)

Methods:

Document create(String token, Document doc, byte[] content)

Create a new document in the repository.

Parameters

- *token* - The session authorization token.
- *doc* - A document object with the new document properties.
- *content* - The document content in bytes.

Returns

- A document object with the properties of the newly created document.

Throws

- *IOException* - An error when inserting document data into the repository.
- *UnsupportedMimeTypeException* - If the uploaded file has an unsupported MIME type.
- *FileSizeExceededException* - If the document content is bigger than the maximum accepted.
- *VirusDetectedException* - If the document is infected by a virus.
- *PathNotFoundException* - If the parent folder doesn't exist.
- *ItemExistsException* - If there is already a document in the repository with the same name.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent document folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Document createSimple(String token, String docPath, byte[] content)

Create a new document in the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The document destination path in the repository.
- *content* - The document content in bytes.

Returns

- A document object with the properties of the newly created document.

Throws

- *IOException* - An error when inserting document data into the repository.
- *UnsupportedMimeTypeException* - If the uploaded file has an unsupported MIME type.
- *FileSizeExceededException* - If the document content is bigger than the maximum accepted.
- *VirusDetectedException* - If the document is infected by a virus.
- *PathNotFoundException* - If the parent folder doesn't exist.
- *ItemExistsException* - If there is already a document in the repository with the same name.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent document folder because of lack of permissions.
    - *RepositoryException* - If there is any general repository problem.

void delete(String token, String docId)

Removes a document from the repository and move it to the user trash.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- none

Throws

- *LockException* - Can't delete a locked document.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void lock(String token, String docPath)

Lock a document, so it is only editable by the locker.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- A Collection of Versions with every document version.

Throws

- *LockException* - If the node is already locked.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: you can't modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void unlock(String token, String docPath)

Unlock a document, so will be editable for other users.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- none

Throws

- *LockException* - If the node is not locked.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Document rename(String token, String docPath, String newName)

Rename a document in the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *newName* - The new document name.

Returns

- Documents object with the new document properties.

Throws

- *PathNotFoundException* - If there is no document in this repository path.
- *ItemExistsException* - If there is already a document in the repository with the same name in the same path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void move(String token, String docPath, String fldPath)

Move a document to another location in the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *fldPath* - The destination folder path.

Returns

- none

Throws

- *PathNotFoundException* - If the fldPath does not exists.
- *ItemExistsException* - If there is already a document in the destination folder with the same name.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document's parent folder or the destination folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Document getProperties(String token, String docPath)

Obtain document properties from the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- The document properties.

Throws

- *PathNotFoundException* - If there is no document in this repository path.
- *RepositoryException* - If there is any general repository problem.

void setProperties(String token, Document doc)

Set the properties of a repository document.

Parameters

- *token* - The session authorization token.
- *doc* - An document object with the properties.

Returns

- none

Throws

- *LockException* - A locked document can't be modified.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem you can't modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void setContent(String token, String docPath, byte[] content)

Set document content in the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *content* - The new document content.

Returns

- none

Throws

- *IOException* - If there is an error setting up the new content.
- *VersionException* - A document checked in can't be modified.
- *LockException* - A locked document can't be modified.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

byte[] getContent(String token, String docPath, boolean checkout)

Obtain document content from the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *checkout* - If the content is retrieved after a check-out or not.

Returns

- none

Throws

- *IOException* - An error when retrieving document data from the repository.
- *PathNotFoundException* - If there is no document in this repository path.
- *RepositoryException* - If there is any general repository problem.

byte[] getContentByVersion(String token, String docPath, String versionId)

Obtain document content from the repository.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *versionId* - The version identification associated to the document content.

Returns

- none

Throws

- *IOException* - An error when retrieving document data from the repository.
- *PathNotFoundException* - If there is no document in this repository path.
- *RepositoryException* - If there is any general repository problem.

void checkout(String token, String docPath)

Checkout the document to edit. The document can't be edited by another user until it is checked in or the checkout is cancelled.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- A Collection with the child documents.

Throws

- *LockException* - A locked document can't be modified.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void cancelCheckout(String token, String docPath)

Cancel previous checked out state in a document.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- A Collection with the child documents.

Throws

- *LockException* - A locked document can't be modified.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Version checkin(String token, String docPath, String comment)

Check in the document to create a new version.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- A version object with the properties of the newly generated version.

Throws

- *LockException* - A locked document can't be modified.
- *VersionException* - If the nodes was not previously checked out.
- *PathNotFoundException* - If there is no document in this repository path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

VersionArray getVersionHistory(String token, String docPath)

Get the document version history.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- A Collection of Versions with every document version.

Throws

- *PathNotFoundException* - If there is no document in this repository path.
- *RepositoryException* - If there is any general repository problem.

void restoreVersion(String token, String docPath, String versionId)

Revert the document to a specific previous version.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.
- *versionId* - The version id to revert to.

Returns

- none

Throws

- *PathNotFoundException* - If there is no document in this repository path.
  - *AccessDeniedException* - If there is any security problem: You cannot modify the document because of lack of permissions.
  - *RepositoryException* - If there is any general repository problem.

DocumentArray getChilds(String token, String fldId)

Retrieve a list of child documents from an existing folder.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.

Returns

- A Collection with the child documents.

Throws

- *PathNotFoundException* - If there is no document in this repository path.
- *RepositoryException* If there is any general repository problem.

boolean isValid(String token, String docPath)

Test if a document path is valid.

Parameters

- *token* - The session authorization token.
- *docPath* - The path that identifies an unique document.

Returns

- True if the path denotes a document, otherwise false.

Throws

- *AccessDeniedException* - If there is any security problem: You cannot access this folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.
- *PathNotFoundException* - If there is no folder in the repository with this path.

String getPath(String token, String uuid)

The document path from a UUID.

Parameters

- *token* - The session authorization token.
- *uuid* - The unique document identifier.

Returns

- The document path or null if this UUID does not correspond to a document node.

Throws

- *AccessDeniedException* - If there is any security problem: You cannot access this folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Usage:

Dot Net Client:

The following is sample code of making this API method call getChilds() from a dot net client:

```
 [System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://endpoint.ws.doccept.com/",
ResponseNamespace="http://endpoint.ws.doccept.com/", Use=System.Web.Services.Description.SoapBindingUse.Literal)]

    [return: System.Xml.Serialization.XmlArrayAttribute("return")]

    [return: System.Xml.Serialization.XmlArrayItemAttribute("item", Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]

    public document[] getChilds(string token, string fldPath) {

      object[] results = this.Invoke("getChilds", new object[] {

            token,

            fldPath});

      return ((document[])(results[0]));}
```

The following is sample code of making this API method call getVersionHistory() from a dot net client:

```
 [System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://endpoint.ws.doccept.com/",
ResponseNamespace="http://endpoint.ws.doccept.com/", Use=System.Web.Services.Description.SoapBindingUse.Literal)]

    [return: System.Xml.Serialization.XmlArrayAttribute("return")]

    [return: System.Xml.Serialization.XmlArrayItemAttribute("item", Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]

    public version[] getVersionHistory(string token, string docPath) {

      object[] results = this.Invoke("getVersionHistory", new object[] {

            token,

            docPath});

      return ((version[])(results[0]));

    }
```

JAVA Client:

The following is sample code of making this API method call from a Java client:

```
@WebService(name = "DOCDocument", serviceName = "DOCDocument", targetNamespace = "http://ws.doccept.com")
public class DocumentService {
        private static Logger log = LoggerFactory.getLogger(DocumentService.class);

        @WebMethod
        public Document create(@WebParam(name = "token") String token,
                        @WebParam(name = "doc") Document doc,
                        @WebParam(name = "content") byte[] content) throws IOException, UnsupportedMimeTypeException,
                        FileSizeExceededException, UserQuotaExceededException, VirusDetectedException,
```

```
                ItemExistsException, PathNotFoundException, AccessDeniedException, RepositoryException,
                DatabaseException, ExtensionException, AutomationException {
        log.debug("create({})", doc);
        DocumentModule dm = ModuleManager.getDocumentModule();
        ByteArrayInputStream bais = new ByteArrayInputStream(content);
        Document newDocument = dm.create(token, doc, bais);
        bais.close();
        log.debug("create: {}", newDocument);
        return newDocument;
    }

    @WebMethod
    public Document createSimple(@WebParam(name = "token") String token,
                @WebParam(name = "docPath") String docPath,
                @WebParam(name = "content") byte[] content) throws IOException, UnsupportedMimeTypeException,
                FileSizeExceededException, UserQuotaExceededException, VirusDetectedException,
                ItemExistsException, PathNotFoundException, AccessDeniedException, RepositoryException,
                DatabaseException, ExtensionException, AutomationException {
        log.debug("createSimple({})", docPath);
        DocumentModule dm = ModuleManager.getDocumentModule();
        ByteArrayInputStream bais = new ByteArrayInputStream(content);
        Document doc = new Document();
        doc.setPath(docPath);
        Document newDocument = dm.create(token, doc, bais);
        bais.close();
        log.debug("createSimple: {}", newDocument);
        return newDocument;
    }
```

RESTful Client:
The following is the RESTful command line calls for DOCAuth API.

To create a document, we need to provide the document binary data:

```
$ curl -u admin:admin -H "Accept: application/json" \

  -X POST -F docPath=/root/newDoc.txt -F content=@newDoc.txt \

  http://localhost:8080/doccept/services/rest/document/createSimple
```

DOCFolder
DOCFolder API provides methods related to folder management such as creating a folder, deleting, renaming, moving, fetching child folders etc.

Summary of API methods:

1	Folder create(String token, Folder fld)
2	Folder getProperties(String token, String fldPath)
3	void delete(String token, String fldPath)
4	Folder rename(String token, String fldPath, String newName)
5	void move(String token, String fldPath, String dstPath)
6	FolderArray getChilds(String token, String fldPath)
7	boolean isValid(String token, String fldPath)
8	String getPath(String token, String uuid)

Methods:

Folder create(String token, Folder fld)

Create a new folder in the repository.

Parameters

- *token* - The session authorization token.
- *fld* - A folder object with the new folder properties.

Returns

- A folder object with the new created folder properties.

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *ItemExistsException* - If there is already a folder in the repository with the same name in the same path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Folder getProperties(String token, String fldPath)

Obtains properties from a previously created folder.

Parameters

- *token* -The session authorization token .
- *fldPath* - The path that identifies an unique folder.

Returns

- A folder object with the selected folder properties.

Throws

- *ItemNotFoundException* - If the indicated folder doesn't exist.
- *RepositoryException* - If there is any general repository problem.

void delete(String token, String fldPath)

Delete a folder from the repository. It is a logical delete, so it is moved to the user trash and can be restored.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.

Returns

- None.

Throws

- *LockException* - Can't delete a folder with locked documents.
- *PathNotFoundException* - If there is no folder in the repository in this path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Folder rename(String token, String fldPath, String newName)

Rename a folder in the repository.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.
- *newName* - The new folder name.

Returns

- A folder object with the new folder properties.

Throws

- *PathNotFoundException* - If there is no folder in the repository in this path
- *ItemExistsException* - If there is already a folder in the repository with the same name in the same path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void move(String token, String fldPath, String dstPath)

Move a folder to another location in the repository.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.
- *dstPath* - The path of the destination folder.

Returns

- A folder object with the new folder properties.

Throws

- *PathNotFoundException* - If the dstPath does not exists.
- *ItemExistsException* - If there is already a folder in the destination folder with the same name.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder or the destination folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

FolderArray getChilds(String token, String fldPath)

Retrieve a list of child folders from an existing one.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.

Returns

- A Collection with the child folders.

Throws

- *PathNotFoundException* - If there is no folder in the repository in this path.
- *RepositoryException* - If there is any general repository problem.

boolean isValid(String token, String fldPath)

Test if a folder path is valid.

Parameters

- *token* - String with user authorization info.
- *fldPath* - The path that identifies an unique folder.

Returns

- True if is a valid folder path, otherwise false.

Throws

- *AccessDeniedException* - If there is any security problem: You cannot access this folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.
- *PathNotFoundException* - If there is no folder in the repository with this path.

String getPath(String token, String uuid)

The folder path from a UUID.

Parameters

- *token* - The session authorization token.
- *uuid* - The unique folder identifier.

Returns

- The folder path or null if this UUID does not correspond to a folder node.

Throws

- *AccessDeniedException* - If there is any security problem: you can't access this folder because of lack of permissions.

- *RepositoryException* - If there is any general repository problem.

Usage:

Dot Net Client:
The following is sample code of making this API method call getChilds() from a dot net client:

```
 [System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://endpoint.ws.doccept.com/",
ResponseNamespace="http://endpoint.ws.doccept.com/", Use=System.Web.Services.Description.SoapBindingUse.Literal)]

    [return: System.Xml.Serialization.XmlArrayAttribute("return")]

    [return: System.Xml.Serialization.XmlArrayItemAttribute("item", Form=System.Xml.Schema.XmlSchemaForm.Unqualified)]

    public document[] getChilds(string token, string fldPath) {

      object[] results = this.Invoke("getChilds", new object[] {

            token,

            fldPath});

      return ((document[])(results[0]));}
```

JAVA Client:
The following is sample code of making this API method call from a JAVA client:

```
@WebService(name = "DOCFolder", serviceName = "DOCFolder", targetNamespace = "http://ws.doccept.com")
public class FolderService {
        private static Logger log = LoggerFactory.getLogger(FolderService.class);

        @WebMethod
        public Folder create(@WebParam(name = "token") String token,
                        @WebParam(name = "fld") Folder fld) throws AccessDeniedException, RepositoryException,
                        PathNotFoundException, ItemExistsException, DatabaseException, ExtensionException, AutomationException {
                log.debug("create({}, {})", token, fld);
                FolderModule fm = ModuleManager.getFolderModule();
                Folder newFolder = fm.create(token, fld);
                log.debug("create: {}", newFolder);
                return newFolder;
        }
```

RESTful Client:
The following are the RESTful command line calls for DOCAuth API.

To create a new folder in the repository:

```
$ curl -u admin:admin -H "Accept: application/json" \

  -X POST -H "Content-Type: application/json" -d '/root/newfolder' \

  http://localhost:8080/doccept/services/rest/folder/createSimple
```

DOCMail

DOCMail API provides methods related to mail management such as creating a mail, deleting, renaming, moving etc.

Summary of API methods:

1	Mail create(String token, Mail mail)
2	Mail getProperties(String token, String mailPath)
3	void delete(String token, String mailPath)
4	Mail rename(String token, String mailPath, String newName)
5	void move(String token, String mailPath, String dstPath)
6	MailArray getChilds(String token, String fldPath)
7	boolean isValid(String token, String mailPath)
8	String getPath(String token, String uuid)

Methods:

Mail create(String token, Mail mail)

Create a new mail in the repository.

Parameters

- *token* - The session authorization token.
- *mail* - A mail object with the new mail properties.

Returns

- A mail object with the newly created mail properties.

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *ItemExistsException* - If there is already a mail in the repository with the same name in the same path.

- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Mail getProperties(String token, String mailPath)

Obtains properties from a previously created mail.

Parameters

- *token* -The session authorization token .
- *mailPath* - The path that identifies an unique mail.

Returns

- A mail object with the selected mail properties.

Throws

- *ItemNotFoundException* - If the indicated mail doesn't exist.
- *RepositoryException* - If there is any general repository problem.

void delete(String token, String mailPath)

Delete a mail from the repository. It is a logical delete, so it is moved to the user trash and can be restored.

Parameters

- *token* - The session authorization token.
- *mailPath* - The path that identifies an unique mail.

Returns

- None

Throws

- *LockException* - Can't delete a mail with locked documents.
- *PathNotFoundException* - If there is no mail in the repository in this path.

- *AccessDeniedException* - If there is any security problem: You cannot modify the mail because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Mail rename(String token, String mailPath, String newName)

Rename a mail in the repository.

Parameters

- *token* - The session authorization token.
- *mailPath* - The path that identifies a mail node.
- *newName* - The new mail name.

Returns

- A mail object with the new mail properties.

Throws

- *PathNotFoundException* - If there is no mail in the repository in this path
- *ItemExistsException* - If there is already a mail in the repository with the same name in the same path.
- *AccessDeniedException* - If there is any security problem: You cannot modify the mail because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void move(String token, String mailPath, String dstPath)

Move a mail to another location in the repository.

Parameters

- *token* - The session authorization token.
- *mailPath* - The path that identifies an unique mail.
- *dstPath* - The path of the destination folder.

Returns

- A mail object with the new mail properties.

Throws

- *PathNotFoundException* - If the dstPath does not exists.
- *ItemExistsException* - If there is already a mail in the destination folder with the same name.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder or the destination folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

MailArray getChilds(String token, String fldPath)

Retrieve a list of child mails from an existing folder.

Parameters

- *token* - The session authorization token.
- *fldPath* - The path that identifies an unique folder.

Returns

- A Collection with the child mails.

Throws

- *PathNotFoundException* - If there is no folder in the repository in this path.
- *RepositoryException* - If there is any general repository problem.

boolean isValid(String token, String mailPath)

Test if a mail path is valid.

Parameters

- *token* - String with user authorization info.
- *mailPath* - The path that identifies an unique mail.

Returns

- True if is a valid mail path, otherwise false.

Throws

- *AccessDeniedException* - If there is any security problem: You cannot access this mail because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.
- *PathNotFoundException* - If there is no mail in the repository with this path.

String getPath(String token, String uuid)

The mail path from a UUID.

Parameters

- *token* - The session authorization token.
- *uuid* - The unique mail identifier.

Returns

- The mail path or null if this UUID does not correspond to a mail node.

Throws

- *AccessDeniedException* - If there is any security problem: You cannot access this folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Usage:

JAVA Client:
The following is sample code of making this API method call from a Java client:

```
@WebService(name = "DOCMail", serviceName = "DOCMail", targetNamespace = "http://ws.doccept.com")
public class MailService {
	private static Logger log = LoggerFactory.getLogger(MailService.class);

	@WebMethod
	public Mail create(@WebParam(name = "token") String token,
				@WebParam(name = "mail") Mail mail) throws PathNotFoundException, ItemExistsException,
				VirusDetectedException, AccessDeniedException, RepositoryException, DatabaseException,
				UserQuotaExceededException {
		log.debug("create({}, {})", token, mail);
		MailModule mm = ModuleManager.getMailModule();
		Mail newMail = mm.create(token, mail);
		log.debug("create: {}", newMail);
		return newMail;
	}
```

DOCProperty
DOCProperty API provides methods related to document properties management such as adding and removing keywords, categories.

Summary of API methods:

1	void addCategory(String token, String nodePath, String catId)
2	void removeCategory(String token, String nodePath, String catId)
3	void addKeyword(String token, String nodePath, String keyword)
4	void removeKeyword(String token, String nodePath, String keyword)

Methods:

void addCategory(String token, String nodePath, String catId)

Add a category to a node.

Parameters

- *token* - The session authorization token.
- *nodePath* - A mail object with the new mail properties.
- *catId* - Category id (the UUID of the category node).

Returns

- None

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void removeCategory(String token, String nodePath, String catId)

Remove a previously category from a document.

Parameters

- *token* - The session authorization token.
- *nodePath* - A mail object with the new mail properties.
- *catId* - Category id (the UUID of the category node).

Returns

- None

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void addKeyword(String token, String nodePath, String keyword)

Add a keyword to a document.

Parameters

- *token* - The session authorization token.
- *nodePath* - A mail object with the new mail properties.
- *keyword* - The keyword to be added.

Returns

- None

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

void removeKeyword(String token, String nodePath, String keyword)

Remove a previously assigned keyword from a document.

Parameters

- *token* - The session authorization token.
- *nodePath* - A mail object with the new mail properties.
- *keyword* - The keyword to be removed.

Returns

- None

Throws

- *PathNotFoundException* - If the parent folder doesn't exist.
- *AccessDeniedException* - If there is any security problem: You cannot modify the parent folder because of lack of permissions.
- *RepositoryException* - If there is any general repository problem.

Usage:

JAVA Client:
The following is sample code of making this API method call from a dot net client:

```
@WebService(name = "DOCProperty", serviceName = "DOCProperty", targetNamespace = "http://ws.doccept.com")
public class PropertyService implements PropertyModule {
	private static Logger log = LoggerFactory.getLogger(PropertyService.class);

	@WebMethod
	public void addCategory(@WebParam(name = "token") String token,
				@WebParam(name = "nodePath") String nodePath,
				@WebParam(name = "catId") String catId) throws VersionException,
				LockException, PathNotFoundException, AccessDeniedException, RepositoryException,
				DatabaseException {
		log.debug("addCategory({}, {}, {})", new Object[] { token, nodePath, catId });
		PropertyModule pm = ModuleManager.getPropertyModule();
		pm.addCategory(token, nodePath, catId);
		log.debug("addCategory: void");
	}
```

DOCSearch

DOCSearch API provides methods related to repository search such as searching based on name, keywords, content, categories etc.

Summary of API calls:

1	QueryResultArray findByContent(String token, String words)
2	QueryResultArray findByName(String token, String words)
3	QueryResultArray findByKeywords(String token, StringArray keywords)
4	QueryResultArray findByStatement(String token, String statement, String type)
5	QueryResultArray find(String token, QueryParams params)
6	IntegerPairArray getKeywordMap(String token, StringArray filter)
7	DocumentArray getCategorizedDocuments(String token, String categoryId)

Methods:

QueryResultArray findByContent(String token, String words)

Search for documents using its indexed content.

Parameters

- *token* - The session authorization token.
- *expression* - Expression to be searched.

Returns

- A collection of document which content matched the searched expression.

Throws

- *RepositoryException* - If there is any general repository problem.

QueryResultArray findByName(String token, String words)

Search for documents by document name.

Parameters

- *token* - The session authorization token.
- *expression* – Expression to be searched.

Returns

- A collection of document which name matched the searched expression.

Throws

- *RepositoryException* - If there is any general repository problem.

QueryResultArray findByKeywords(String token, StringArray keywords)

Search for documents using its associated keywords.

Parameters

- *token* - The session authorization token.
- *keywords* - Keywords to be used to filter the search.

Returns

- A collection of document which keywords matched the searched expression.

Throws

- *RepositoryException* - If there is any general repository problem.

QueryResultArray findByStatement(String token, String statement, String type)

Search for documents and folder nodes specifying a complex query statement.

Parameters

- *token* - The session authorization token.
- *statement* - Query statement to be executed.
- *type* - The query language can be "sql" or "xpath".

Returns

- A collection of document from the resulting query statement.

Throws

- *RepositoryException* - If there is any general repository problem or the query fails.

QueryResultArray find(String token, QueryParams params)

Performs a complex search by content, name and keywords (between others).

Parameters

- *token* - The session authorization token.
- *params* - The complex search elements.

Returns

- A collection of documents.

Throws

- *RepositoryException* - If there is any general repository problem.
- *IOException* - If something fails when parsing metadata.

IntegerPairArray getKeywordMap(String token, StringArray filter)

Return a Keyword map. This is a hash with the keywords and the occurrence.

Parameters

- *token* - The session authorization token.
- *filter* - A collection of keywords used to obtain the related document keywords.

Returns

- The keyword map.

Throws

- *RepositoryException* - If there is any general repository problem or the query fails.

DocumentArray getCategorizedDocuments(String token, String categoryId)

Get the documents within a category

Parameters

- *token* - The session authorization token
- *categoryId* - The category id (UUID)

Returns

- A Collection of documents in the category

Throws

- *RepositoryException* - If there is any general repository problem or the query fails.

Usage:

Dot Net Client:
The following is sample code of making this API method call from a dot net client:

```
[System.Web.Services.Protocols.SoapRpcMethodAttribute("", RequestNamespace="http://endpoint.ws.doccept.com/",
ResponseNamespace="http://endpoint.ws.doccept.com/", Use=System.Web.Services.Description.SoapBindingUse.Literal)]
    public void deleteSearch(string token, int qpId) {
      this.Invoke("deleteSearch", new object[] {
                token,
                qpId});
    }
```

JAVA Client:

The following is example code of Search function in Doccept (Java based client for Doccept web services):

```
@WebService(name = "DOCSearch", serviceName = "DOCSearch", targetNamespace = "http://ws.doccept.com")
public class SearchService {
        private static Logger log = LoggerFactory.getLogger(SearchService.class);
```

```
@WebMethod
	public Document[] getCategorizedDocuments(@WebParam(name = "token") String token,
			@WebParam(name = "categoryId") String categoryId) throws RepositoryException, DatabaseException {
		log.debug("getCategorizedDocuments({}, {})", token, categoryId);
		SearchModule sm = ModuleManager.getSearchModule();
		List<Document> col = sm.getCategorizedDocuments(token, categoryId);
		Document[] result = (Document[]) col.toArray(new Document[col.size()]);
		log.debug("getCategorizedDocuments: {}", result);
		return result;
	}
```

```
@WebService(name = "DOCSearch", serviceName = "DOCSearch", targetNamespace = "http://ws.doccept.com")
public class SearchService {
	private static Logger log = LoggerFactory.getLogger(SearchService.class);

	@WebMethod
	public QueryResult[] findByContent(@WebParam(name = "token") String token,
			@WebParam(name = "content") String content) throws IOException, ParseException,
			RepositoryException, DatabaseException {
		log.debug("findByContent({}, {})", token, content);
		SearchModule sm = ModuleManager.getSearchModule();
		List<QueryResult> col = sm.findByContent(token, content);
		QueryResult[] result = (QueryResult[]) col.toArray(new QueryResult[col.size()]);
		log.debug("findByContent: {}", result);
		return result;
	}
```

RESTful Client:

The following are the RESTful command line calls for DOCSearch API.

Search simply by content:

```
$ curl -u amin:admin -H "Accept: application/json" -X GET \
  http://localhost:8080/doccept/services/rest/search/findByContent?content=santo+grial
```

Search by keyword:

```
 $ curl -u admin:admin -H "Accept: application/json" -X GET \
```

```
http://localhost:8080/doccept/services/rest/search/findByKeywords?keyword=santo\&keyword=grial
```