Project: DataUp API Guide

Version 1.0 - August, 2012

Abstract

This document describes the API's which are exposed as part of DataUp Project.

For updates to this document and the rest of the DataUp Project DataUp, see dataup.cdlib.org

Contents

API

Retrieve Repositories

This web method retrieves all repositories which are created in DataUp.

Method Name: **Repositories**

Retrieve User Agreement

This web method gets the user agreement text for the specified Repository Name

Method Name: **GetUserAgreement**

Parameters:

- `RepositoryName`: Name of the repository for which agreement text has to be retrieved.

Sample Implementation:

```
using (WebClient client = new WebClient())
{
    NameValueCollection values = new NameValueCollection();
    values.Add("RepositoryName", repositoryName);

    byte[] response = client.UploadValues(new Uri(Constants.DataUpWebSer
viceLink + "/GetUserAgreement"), "POST", values);

    string responseString = Encoding.UTF8.GetString(response);

    if (!string.IsNullOrWhiteSpace(responseString))
    {
        var getResponse = XElement.Parse(responseString);
        agreement = getResponse.Value;
    }
}
```

Get Identifier

This web method gets the identifier from the repository specified in the repository model instance.

Method Name: **GetIdentifier**

Parameters:

- `QueryData:[Base64Encoded]`: This is of type `<DQueryData>`. This parameter contains the data which is required for the repository to get the identifier.
- `RepositoryModel:[Base64Encoded]`: This is of type `<DRepositoryBase>`. This parameter contains the following data.
    i. Repository Name: Name of the repository.

ii. Authorization Token: The authentication token to authenticate the user who is accessing the web method

Sample Implementation:

```
DQueryData queryData = new DQueryData();
List<DKeyValuePair<string, string>> kayValue = new List<DKeyValuePair<st
ring, string>>()
{
    new DKeyValuePair<string, string>("Profile", "dcxl_ark_only"),
    new DKeyValuePair<string, string>("Who", "Publisher"),
    new DKeyValuePair<string, string>("What", "Title"),
    new DKeyValuePair<string, string>("When", "2012-12-12")
};

queryData.KeyValuePair = kayValue.ToArray();

DRepositoryBase repositoryBase = new DRepositoryBase()
{
    RepositoryName = "RepositoryName",
    Authorization = Convert.ToBase64String(System.Text.UTF8Encoding.UTF8
.GetBytes("Username" + ":" + "Password"))
};

string queryDataString = queryData.SerializeObject<DQueryData>("queryDat
a");
string repositoryDataString = repositoryBase.SerializeObject<DRepository
Base>("repositoryModel");

using (WebClient client = new WebClient())
{
    NameValueCollection values = new NameValueCollection();
    values.Add("QueryData", queryDataString.EncodeTo64());
    values.Add("RepositoryModel", repositoryDataString.EncodeTo64());
    byte[] response = client.UploadValues(new Uri(Constants.DataUpWebSer
viceLink + "/GetIdentifier"), "POST", values);

    string responseString = Encoding.UTF8.GetString(response);

    if (!string.IsNullOrWhiteSpace(responseString))
    {
        var getIdentifierElement = XElement.Parse(responseString);
        identifier = getIdentifierElement.Value;
    }
}
```

Publish File

This web method publishes the file to repository.

Method Name: **UploadFile**

Parameters:

- `QueryData:[Base64Encoded]`: This is of type `<DQueryData>`. This parameter contains the data which is required for the repository to get the identifier.
- `RepositoryModel:[Base64Encoded]`: This is of type `<DRepositoryBase>`. This parameter contains the following data.
    i. Repository Name: Name of the repository.
    ii. Authorization Token: The authentication token to authenticate the user who is accessing the web method
- `File:[Base64Encoded]`: This is of type `<DFile>`. This parameter contains the following data.
    i. File Name: Name of the file which is being published.
    ii. File Extension: extension of the file which is being published
    iii. File Content: Content of the file as byte[]

Sample Implementation:

```
DQueryData queryData = new DQueryData();
queryData.MetadataTypeID = metadataList.MetadataTypeID;
List<DKeyValuePair<string, string>> content = new List<DKeyValuePair<str
ing, string>>();
foreach (var item in metadataList.Metadata)
{
    DKeyValuePair<string, string> metadata = new DKeyValuePair<string, s
tring>();
    metadata.Key = item.Name;
    metadata.Value = item.Value;
    content.Add(metadata);
}

content.Add(new DKeyValuePair<string, string>() { Key = "Profile", Value
 = "dcxl_ark_only" });
content.Add(new DKeyValuePair<string, string>() { Key = "who", Value = "
publisher" });
content.Add(new DKeyValuePair<string, string>() { Key = "what", Value =
"title" });
content.Add(new DKeyValuePair<string, string>() { Key = "when", Value =
 "2012-02-02" });
content.Add(new DKeyValuePair<string, string>() { Key = "where", Value =
 "identifier" });
content.Add(new DKeyValuePair<string, string>() { Key = "ARK", Value = "
identifier" });

queryData.KeyValuePair = content.ToArray();

List<DataUp.Web.Service.Model.ParameterMetadataDetail> parameters = new
List<DataUp.Web.Service.Model.ParameterMetadataDetail>();
foreach (var item in metadataList.ParamterMetadata)
{
    parameters.Add(new DataUp.Web.Service.Model.ParameterMetadataDetail(
)
    {
        Name = item.Name,
        Description = item.Description,
```

```
        Type = item.Type,
        Units = item.Units
    });
}
queryData.ParameterMetadata = parameters.ToArray();

DRepositoryBase repositoryBase = new DRepositoryBase()
{
    RepositoryName = details.RepositoryName,
    Authorization = details.Authorization
};

DFile file = new DFile();
file.isCompressed = details.IsCompressed;

fileStream = new MemoryStream();
updatedDocument.DataStream.CopyToStream(fileStream);
if (!details.IsCompressed)
{
    file.FileName = Path.GetFileName(details.FileName);
    file.FileExtentsion = Path.GetExtension(details.FileName);
    file.FileContent = fileStream.GetBytes();
}
else
{
    file.FileName = Path.GetFileNameWithoutExtension(details.FileName) +
 ".zip";
    file.FileExtentsion = ".zip";

    // TODO: Integrated code for converting to csv
    using (SpreadsheetDocument excelDocument = SpreadsheetDocument.Open(
fileStream, true))
    {
        var zipFileStream = excelDocument.GetZipFileStream(Path.GetFileN
ameWithoutExtension(details.FileName));
        if (zipFileStream != null)
        {
            file.FileContent = zipFileStream.GetBytes();
        }
    }
}
string queryDataString = queryData.SerializeObject<DQueryData>("queryDat
a");
string repositoryDataString = repositoryBase.SerializeObject<DRepository
Base>("repositoryModel");
string queryFIle = file.SerializeObject<DFile>("file");

using (WebClient client = new WebClient())
{
    NameValueCollection values = new NameValueCollection();
    values.Add("QueryData", queryDataString.EncodeTo64());
    values.Add("RepositoryModel", repositoryDataString.EncodeTo64());
    values.Add("File", queryFIle.EncodeTo64());
```

```
    byte[] response = client.UploadValues(new Uri(Constants.DataUpWebSer
viceLink + "/UploadFile"), "POST", values);

    string responseString = Encoding.UTF8.GetString(response);

    if (!string.IsNullOrWhiteSpace(responseString))
    {
        var element = XElement.Parse(responseString);
        if (!element.Value.Contains("false|"))
        {
            returnString = "Your data file was submitted successfully. Y
ou will receive an email shortly with your unique identifier and submiss
ion information.";
        }
        else
        {
            return element.Value;
        }
    }

}
```