AUTODESK **UNIVERSITY** 2013

Programming Autodesk PLM 360 Using REST

Doug Redmond – Autodesk

DV1518

The PLM 360 API is here. Learn the basics of this REST-ful API to take your processes to the next level. Build enterprise-level integrations, desktop apps, and mobile apps. The possibilities are endless.

Learning Objectives

At the end of this class, you will be able to:

- Describe the Autodesk PLM 360 API framework
- Use the basics of representational state transfer (REST)
- Authenticate using Oxygen
- Understand the basic feature set of the REST API

About the Speaker

Doug Redmond has 9+ years of experience developing and using the Vault API. He is now the principle engineer for the REST API in PLM 360.

He also runs It's All Just Ones and Zeros, a blog for Vault and PLM 360 development.

Introduction

This document is a supplement to the slide deck and presentation. It will go deeper into topics covered in the sides. It will also cover additional topics that were not able to be presented due to time constraints.

Useful Links

Wikipedia page on REST
http://en.wikipedia.org/wiki/REST_API
A good starting point if you have never used a REST API.

OAuth 1.0 Spec
http://tools.ietf.org/html/rfc5849
If you are still trying to understand just what OAuth is, I suggest having a look at the spec.

DotNetOpenAuth
http://dotnetopenauth.net/
Microsoft does not yet have a .Net library for OAuth. This is the best third-party library I've found for .Net.

PLM 360 Online Help
http://help.autodesk.com/view/PLM/ENU/
This is the official help for PLM 360. It contains a section on the REST API, which documents all the supported REST URLs.

It's All just Ones and Zeros
http://justonesandzeros.typepad.com/
This is an official Autodesk blog specific to the PLM 360 and Vault API's. Updates weekly with a sample app posted every month. Lots of source code examples. Important API announcements are posted here.

PLM 360 Discussion Group
http://forums.autodesk.com/t5/PLM-360-General/bd-p/705
This is the Autodesk-hosted discussion group for all PLM 360 related topics, including the API. It's actively monitored by the PLM 360 team. If you get stuck, this is the place to go to get help.

ADN GitHub
https://github.com/ADN-DevTech
This is where the Autodesk Developer Network posts their sample apps. There is a section of OAuth examples and a section of PLM 360 examples.

Binary Data Transfer

For the most part, the PLM 360 REST API is straightforward and easy to use. However, if binary data is involved, the steps are more confusing. Below are descriptions of all the calls that currently involve binary data. See the **additional materials**, for code samples of each call.

Some knowledge of HTTP is required. The good news is that HTTP is a well-defined standard and there are lots of libraries for you to use.

Binary data is usually transferred via an **octet stream** or **multipart** HTTP request. Octet stream is where the entire HTTP body is binary data. Multipart is where there are several parts to the request. Each part has its own media type, header, and body. So that makes it possible to mix text and binary data in a single call. For PLM 360, usually the first part is JSON text data and the other parts are binary.

File Create

This is a POST call to **/api/v2/workspaces/{workspaceId}/items/{itemId}/files**. It's a multipart request.
Part 1: A FileUploadRequest object, serialized to JSON
Part 2: The binary contents of the file.

The function returns all File objects on the Item, including the one you just uploaded. You can use the title to locate the new File, since File titles are required to be unique for a given Item.

File Download

This is a GET call to **/api/v2/workspaces/{workspaceId}/items/{itemId}/files/{fileId}**. The behavior of this URL is different depending on the accept type.

If the HTTP request says that octet stream is accepted, then the return body will contain the binary contents of the file. Otherwise, the return body will contain the File object, which is just meta-data about the file.

So if you want to get both the file and the meta-data, you need to make two separate calls to PLM 360.

File Checkin

This is a POST call to **/api/v2/workspaces/{workspaceId}/items/{itemId}/files/{fileId}/checkin**. It's a multipart request. It's very similar to uploading a file. The request involves 2 parts.
Part 1: A FileUploadRequest object, serialized to JSON
Part 2: The binary contents of the file.

The function returns just the updated File object.

Items Create

Adding a new item is a POST call to **/api/v2/workspaces/{workspaceId}/items**. If your item just involves text data, then this can be simple JSON content. However, if there are binary field values you want to set, then this needs to be a multipart call. The common case is an image field.
Part 1: An array of ItemDetail objects.
Parts 2-n: The contents of a single binary field.

This URL lets you create multiple items in a single call. And a given item may have multiple binary fields. So that means PLM 360 needs a way to match each part with each field value. This is done by setting the Content Disposition (RFC 2183) on each part header.
Required values on Content Disposition:

- **disposition-type** – I suggest putting * for this value.
- **filename** – The name of the file. For image fields, PLM 360 uses the extension to determine the image type.
- **size** – The size of the field data, in bytes.
- **index** – The entry in the ItemDetail array that this part corresponds to.
- **key** – The field definition ID that this part corresponds to.

"Key" and "index" are PLM 360 specific values. The other values are defined in the Content Disposition spec.

The function returns a collection of ItemDetail objects representing the newly created PLM 360 objects. All the text fields are included in the return value, but the binary data is not.

Items Update

Multiple items can be updated in a single call. The call is almost exactly the same as creating items except that it's a POST call to **/api/v2/workspaces/{workspaceId}/items**. A multipart call is still required and the content dispositions settings are the same.

Item Update

When updating a single item, do a POST call to **/api/v2/workspaces/{workspaceId}/items/{itemId}**. The rules for binary data are mostly the same as with the create URL. You will still need to do a multi-part request, but you are only dealing with a single Item.
Part 1: The ItemDetail object with updated fields (non-binary).
Parts 2-n: The contents of a single binary field to update.

Just like with creating the item, each binary part needs a Content Disposition header. The only difference is that "index" is not needed since there is only one item.

Required values on Content Disposition:

- **disposition-type** – I suggest putting * for this value.
- **filename** – The name of the file. For image fields, PLM 360 uses the extension to determine the image type.
- **size** – The size of the field data, in bytes.
- **key** – The field definition ID that this part corresponds to.

If you are not updating a binary field, you do not need to use a multipart call. For example, if you are changing the NAME field of an item but are leaving the IMAGE field as-is, then you can just do a single part call containing the ItemDetail.

Item Field Read

The ItemDetail object only contains non-binary field data. If you want the value of a binary field, such as an image, you need to make a separate GET call to **/api/v2/workspaces/{workspaceId}/item-fields/{fieldDefinitionId}/values**. The body of the result will be the binary contents of the field. The media type in the response header will tell you the MIME type for the data. For example a JPEG image will come back with "image/jpeg" as the media type.

A binary field will show up in the ItemDetail fields. However, the value will just be the REST URL for download the binary contents. For example...

```
{ "id" : 2890,
   ...
  "fields" : {
    "NAME" : "Update Image Test",
    "CITY" : null,
    "IMAGE" :
"https://mytenant.autodeskplm360.net/api/v2/workspaces/52/items/2890/field-
values/IMAGE"}}
```

Login Workflows

The authentication instructions for the PLM 360 API are mostly focused on the scenario where the application has UI and a user is typing in their credentials. However, you may find that you have needs outside that scenario. This section will outline some other workflows that may be of use.

At the heart of these workflows is three-legged OAuth 1.0, which is heavily built around the concept of a user, using a web browser, which redirects between different web sites. It's a limited scenario, and becomes difficult to work with when you start introducing concepts like integrations. For example, OAuth 1.0 has no way of passing in username and password. The intent is that the user enters that data directly on the security provider website. Your app only knows about things like access tokens.

OAuth 2.0 is meant to handle some of these cases, but Oxygen does not yet support OAuth 2.0 at the time of this writing.

Auto-login

It can be tedious for the user to log in each time, especially if the user is you and you are debugging your application. Even though you can't pass in username/password data, you can build an auto-login feature.

Basic steps:

1. For the first time run, the user logs in the normal way.
2. After entering the username/password on Oxygen, your app can get the Access Token and Access Token Secret, which are good for 2 days.
3. Store the Access Tokens is a safe place where the app can read it.
4. The next time the app is run, use the stored Access Tokens to skip most of the authentication steps. You can call the "oxygen-login" REST endpoint directly without requiring the user to login again.

The Access Token should be stored in a secure location. Although it's more secure than username/password data, a compromised Access Token could be used maliciously to read and update PLM 360 data.

Renewing The Access Token

If you have a service that runs constantly in the background, like an integration, then 2 days is not long enough for the access token to stay valid. You don't want to force a user to have to re-log in every 2 days just to keep a service running. The solution is to renew the token frequently. Although the token itself can be used for 2 days, it can be renewed up to 14 days after it was issued. Here is how you renew an access token...

1. Read the Session Handle

When you are first issued the access token, one of the pieces of information that comes back is the session handle. Remember this handle along with the token and secret.

Example Response:

```
oauth_token=KLe2eC3792uDhz1rnznFVjr9ibI%3D
&oauth_token_secret=7nxoZ0MXDIdx25xjqIRnMLgzvCs%3D
&oauth_session_handle=CuXWXi9fQ4JIKs%2FxksFdeWLqDCs%3D
&oauth_authorization_expires_in=1209599
&oauth_expires_in=172799
&x_oauth_user_name=Doe.J
&x_oauth_user_guid=200815661123455
&x_consumerscope=https%253A%252F%252Fmysite.autodeskplm360.net
```

(*line breaks are for display only*)

2. Call AccessToken

When you want to renew your token, you will again call POST on https://accounts.autodesk.com/**OAuth/AccessToken**, but the input parameters are a bit different. The oauth_token should be the current access token and the oauth_session_handle should be passed in too. The request should be signed by your Consumer Secret and Access Secret.

Example Request:

```
oauth_verifier=BHePF8wenS
oauth_nonce=d5baf3f6-5c2c-461e-8340-26f885c80640
oauth_version=1.0
oauth_signature_method=HMAC-SHA1
oauth_consumer_key=213f6e53-19fa-4b22-9bfb-5215fbfa7a93
oauth_token=36+dZ74Ig72QxyGtCA8mGJKrCxM=
oauth_timestamp=1375903880
oauth_signature=XK45m8XQE9MLVOz734VDwzbb6Ox
oauth_session_handle=wr00U6DPDT8sDgiqYspXw4oIp/s=
```

The response looks the same as in part one, but there are now new values for the Access Token, Access Secret and Session Handle. The old values are not invalid and cannot be used to access PLM 360.

The timer has been reset so the new Access Token is good for another 2 days and can be renewed up to 14 days later. An automated service can continue renewing tokens indefinitely.

Invalidating The Access Token

Maybe you don't want the Access Token to stay active for 2 days. If you want to immediately invalidate it, make a POST call https://accounts.autodesk.com/**OAuth/InvalidateToken**. As usual, this must be a signed request (use your Consumer Secret and Access Secret), with a bunch of oauth parameters passed in. It also requires that you know the session handle from the last /OAuth/AccessToken call.

Example Request:

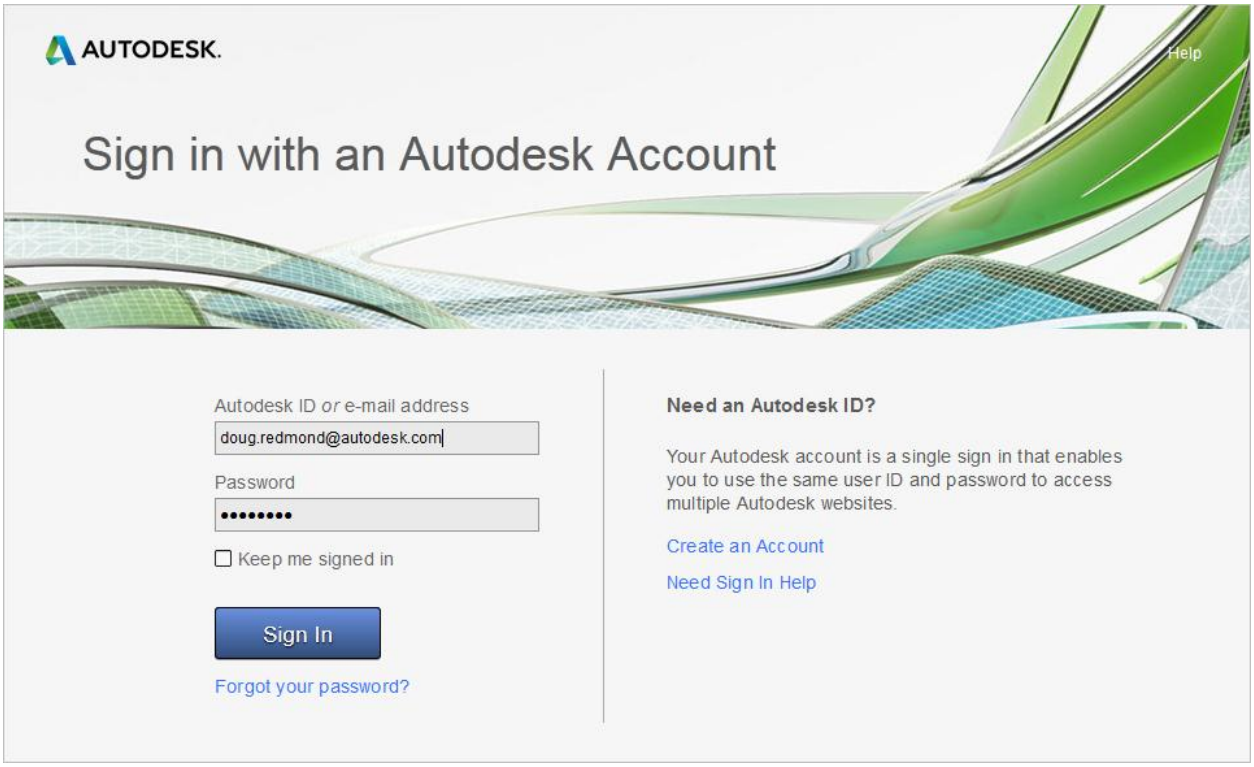
```
oauth_verifier=BHePF8wenS
oauth_nonce=d5baf3f6-5c2c-461e-8340-26f885c80640
oauth_version=1.0
oauth_signature_method=HMAC-SHA1
oauth_consumer_key=213f6e53-19fa-4b22-9bfb-5215fbfa7a93
oauth_token=36+dZ74Ig72QxyGtCA8mGJKrCxM=
oauth_timestamp=1375903880
oauth_signature=XK45m8XQE9MLVOz734VDwzbb6Ox
oauth_session_handle=wr00U6DPDT8sDgiqYspXw4oIp/s=
```

If the call is successful, the Access Token, Access Secret and Session Handle are invalid and cannot be renewed.

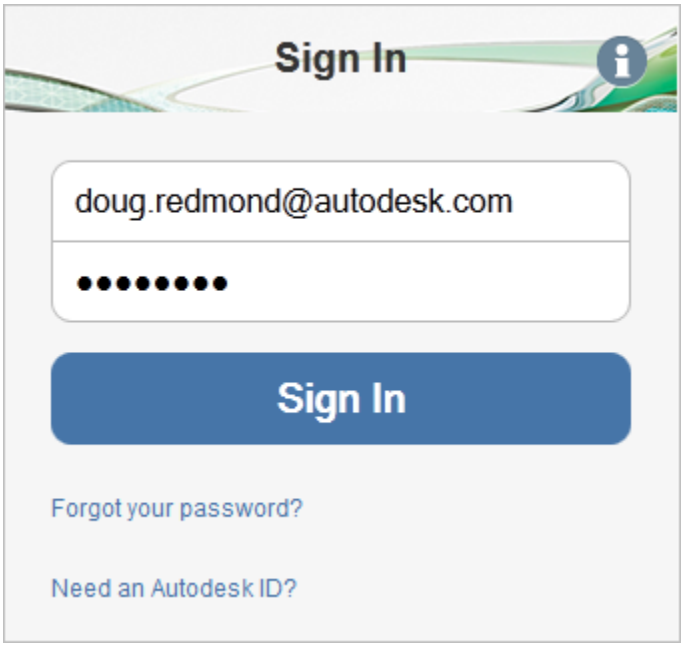
Login Views

There are several views of the login page (https://accounts.autodesk.com/OAuth/Authorize) that you can use depending on your purpose.

The default view is geared for web browsers.



If you want a view designed for mobile apps, add “&viewmode=mobile” to the query string.



Lastly, there is the view for desktop apps. Add “&viewmode=mobile” to the query string to get this view.