Programming Autodesk PLM 360 Using REST

Doug Redmond
Software Engineer, Autodesk

Introduction

This class will show you how to write your own client applications for PLM 360.

This is not a class on scripting.

Introduction

Who knows how to program?

Who has used PLM 360?

Who has used the PLM 360 REST API?

Who has used the Vault API?

Agenda

- Introduction
- PLM 360 Overview
- Introduction to REST
- Authentication
- API features
- Wrapup

About the speaker

Doug Redmond has 9+ years 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.

PLM 360 Overview

PLM 360 Overview

- PLM = Product Lifecycle Management
- In the cloud
- Updates about once a month
- Autodesk single-sign-on used to authenticate (Oxygen)
- Licensing is based on number of active users

PLM 360 Overview

- Demo

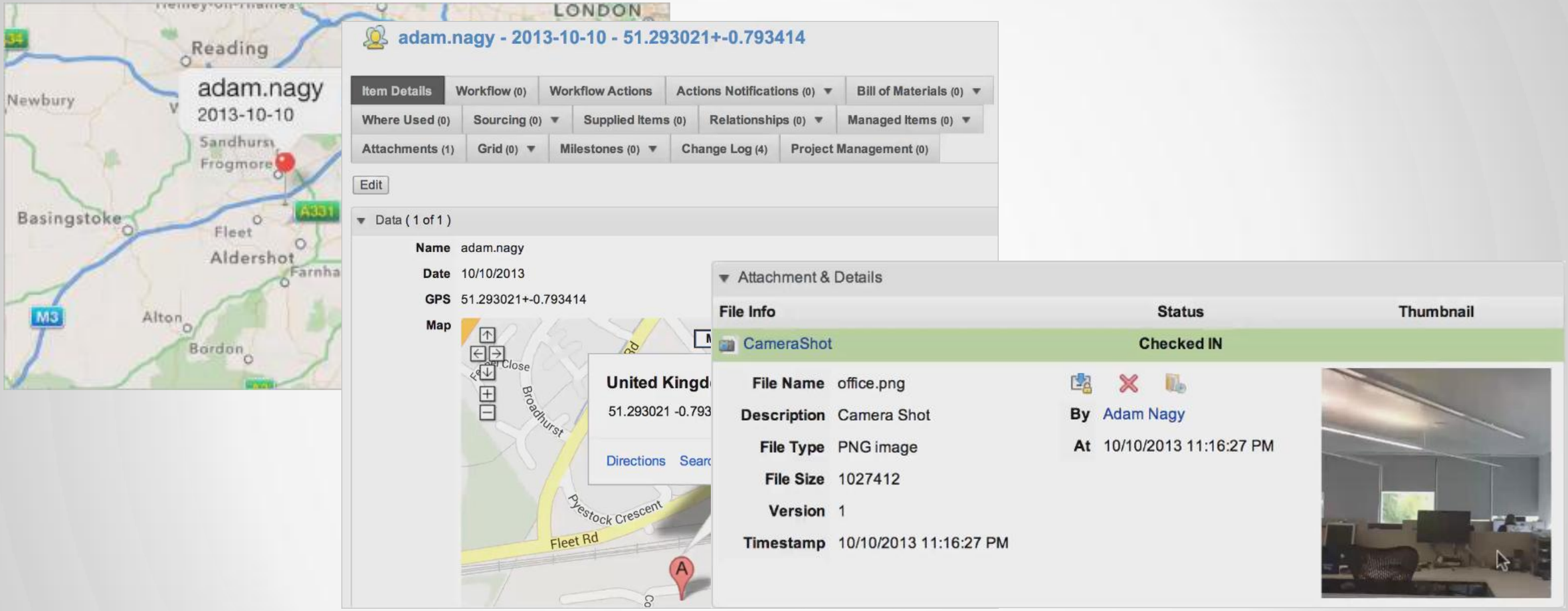
PLM 360 Overview

REST API provides client access to PLM 360.

Uses include:

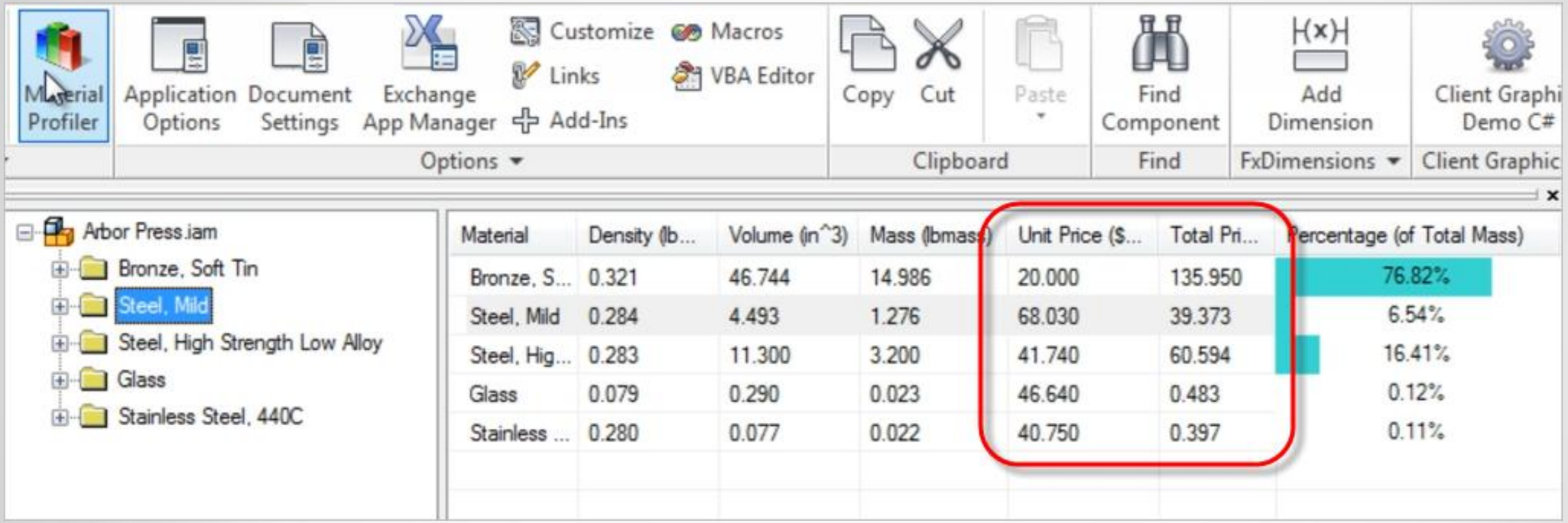
- Desktop apps
- Mobile apps
- Plug-ins
- Integrations

API Examples – GPS Tracker



Source code available from [ADN GitHub]

API Examples – Material Profiler



Source code available from [ADN GitHub]

Introduction to REST

Introduction to REST

How many of you are familiar with a RESTful API?

REST = Representational State Transfer

Dissertation (http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

Implementation Pattern (http://en.wikipedia.org/wiki/REST_API)

Commonly used definition

Introduction to REST

REST uses HTTP features to implement the API.

Concepts:

- HTTP Verbs
- Headers
- URLs and Query Strings
- Content types

Introduction to REST

The URL is the function name.

Traditional API:

ItemDetail item = getItem(int workspaceId, int itemId);

REST API:

GET http://hostname/api/v2/workspaces/{workspaceId}/items/{itemId}

Introduction to REST

URLs attempt to follow CRUD model.
(Create, Read, Update, Delete)

Operation	HTTP Verb
Create	POST
Read	GET
Update	PUT
Delete	DELETE
Other	POST

Everyone confuses these

Introduction to REST

- URL describes the resource being acted on.
  - GET /api/v2/workspaces/12/items
- URL specifies a single resource VS a collection.
  - GET /api/v2/workspaces
  - GET /api/v2/workspaces/12
- URL can describe a hierarchy and/or context.
  - GET /api/v2/workspaces/12/items

Introduction to REST

HTTP allows inputs from multiple places:

- HTTP Verb
- URL
- Query String
- HTTP Header
  - Cookies
  - Content Type
- HTTP Body

PLM Specific REST

Non CRUD operations:

POST /api/v2/workspaces/{workspaceId}/items/{itemId}/files/{fileId}/checkout

POST /api/v2/workspaces/{workspaceId}/items/query

Is it CRUD??:

DELETE /api/v2/workspaces/{workspaceId}/items/{itemId}/files/{fileId}/checkout

PLM Specific REST

- PLM 360 API body only supports JSON
  - JavaScript Object Notation


```
{
  "id" : 2847,
  "rootId" : 2847,
  "version" : 1,
  "revision" : null,
  "workspaceId" : 2,
  "url" : "https://adskredmondd.autodeskplm360.net/api/v2/workspaces/2/items/2847",
  "itemDescriptor" : "TK000001 - Update Mechanical Design",
  "deleted" : false,
  "isWorkingVersion" : true,
  "isLatestVersion" : true,
  "fields" : {
    "TASK_ID" : "TK000001",
    "TITLE" : "Update Mechanical Design",
"picklistFields" : {
    "STATUS" : [ {
      "id" : null,
      "displayName" : "1 - On Track",
      "itemUrl" : null
    } ],
  }
}
```

Notes on JSON

No class names

```
{
  "page" : null,
  "elements" : [ {
    "id" : 52,
    "url" : "https://adskredmondd.autodeskplm360.net/api/v2/workspaces/52",
    "displayName" : "Dougspace",
    "systemName" : "WS_DOUGSPACE",
    "description" : "",
    "workspaceTypeId" : "BASIC"
  } ]
}
```

Can't infer type info from null

Can't infer type on strings

Example JSON is not sufficient documentation on its own.

Demo – GET Through a Web Browser

Log in to PLM 360 and type in a GET URL.

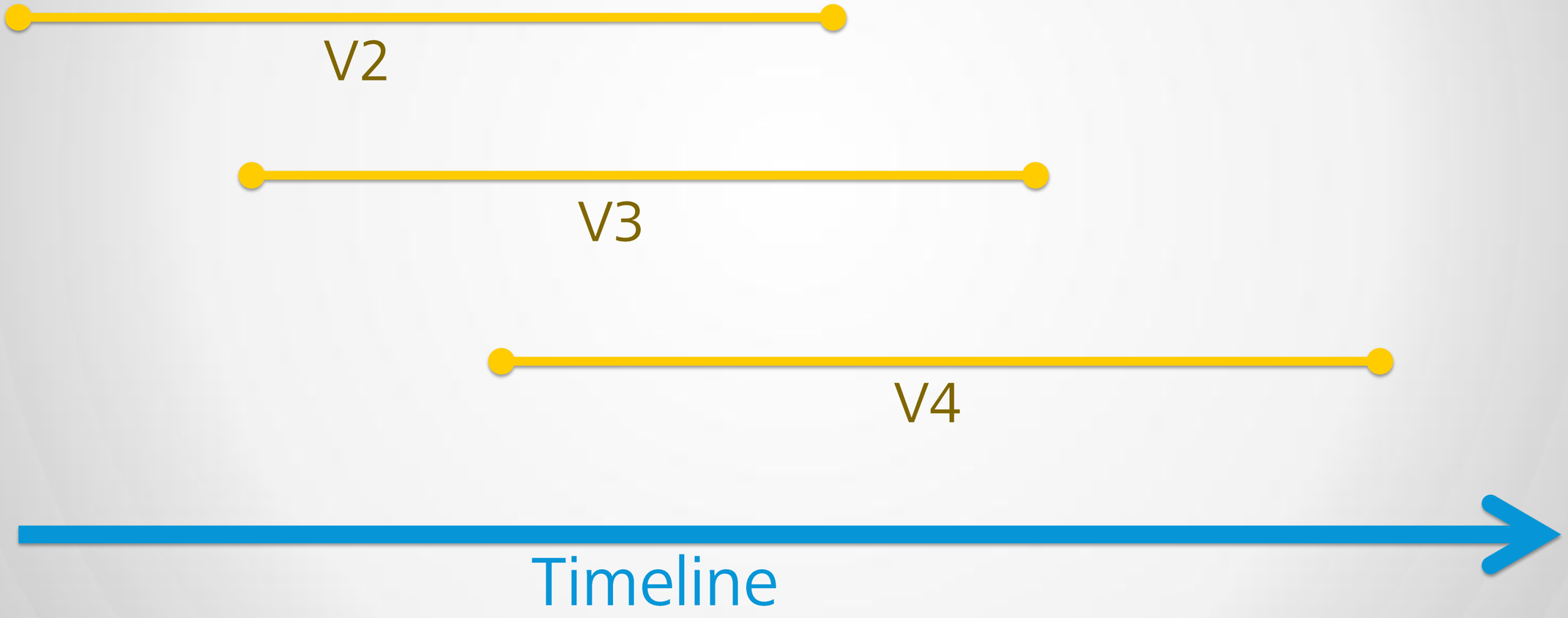
API Versioning

- The officially supported API starts at version 2.
  /api/v2/workspaces/{workspaceId}/items/query

- V1 is for internal use and is unsupported by the Autodesk Developer Network.

API Versioning

APIs will live in parallel.



API Versioning

- V2 will be updated with every PLM release.
  - It will still be called v2.
  - It will still be compatible.*
- V3 will be introduced when compatibility cannot be maintained.
- End of life for an API version will be announced in advance.

* V2 is currently in tech preview mode

Compatibility

Quiz: A product has updated. Is the API still compatible.

Before: void UpdateName(String, String);
After: void UpdateName(Int, String);

Answer: No

Compatibility

Quiz: A product has updated. Is the API still compatible.

Before: void UpdateName(String, String);
After: void UpdateName(String, String);

Before: void UpdateName(String lastName, String firstName);
After: void UpdateName(String firstName, String lastName);

Answer: No

Compatibility

Quiz: A product has updated. Is the API still compatible.

Before: void UpdateName(String firstName, String lastName);
After: void UpdateName(String firstName, String lastName);

Answer: Don't Know. Parameter names don't impact functionality.

Compatibility

My definition of Compatibility:

An API is compatible if your app still works as expected.

Forward Compatibility

Changes to PLM will affect the API. Even if the API schema doesn't change.

/api/v2/workspace-types/

- Currently returns 6 results.
- May return 7 results in a future release.
- Client apps should gracefully handle this situation.

Best Practices

Account for allowed API schema changes:

- Additional endpoints may be added.
- Existing classes may acquire additional properties.
- Enums may acquire additional values.
- The retuned error code might change for a given operation.

Best Practices

- Avoid hacks
- Use nullable types
- Use strings instead of enums in PLM classes
- Log out when your app is completed

Sanity Check

Take a video of your working app.

A lot of screenshots also works.

When PLM updates, you will have a record of how your app used to work.

Authentication

Oxygen

- Autodesk 360 products use an service called Oxygen to authenticate users.
- Oxygen is based on OAuth.

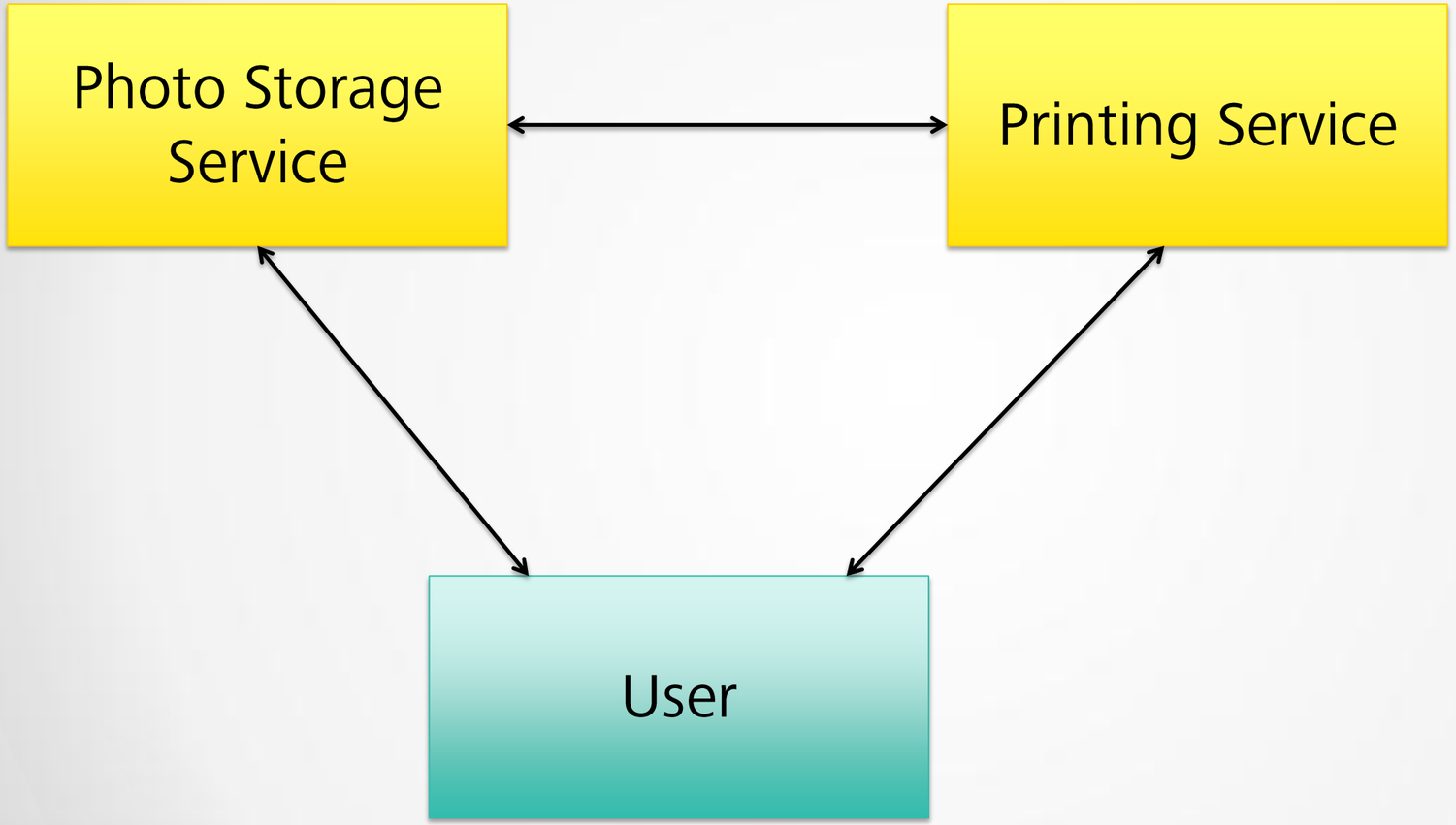
Oxygen

How many people are familiar with OAuth?

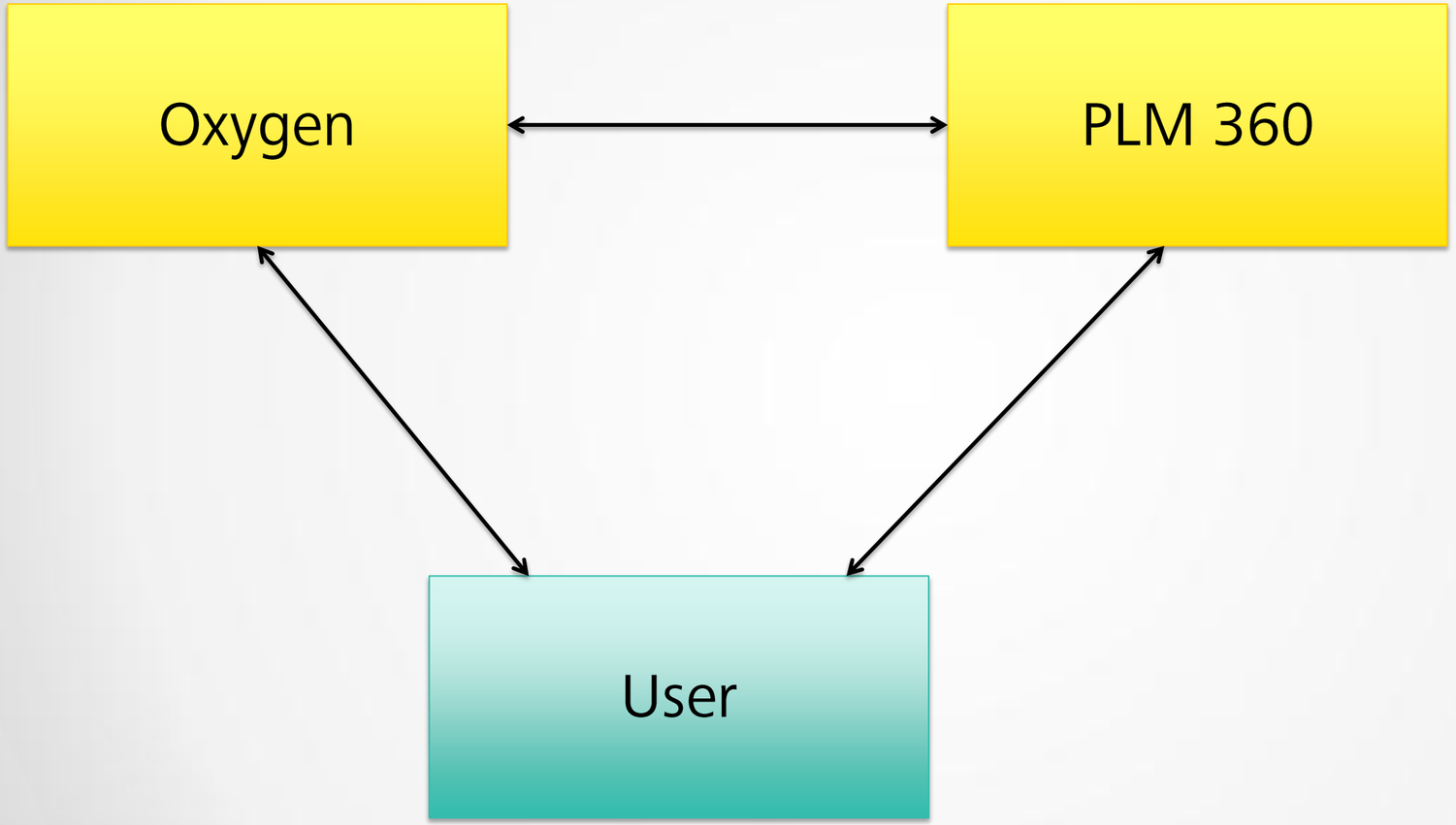
OAuth

- OAuth is a system where resources can be shared securely between separate systems on the web.
- 3-legged OAuth involves 2 services and the user.

OAuth



OAuth



Oauth Concepts

- Public Keys and Secret Keys
  - Public keys get passed between systems.
  - Private keys should not get passed around. They are used for signing requests.

OAuth Concepts – Tokens

- Consumer tokens
  - Identifies an application
- Request tokens
  - Used to make OAuth calls before logging in.
- Access tokens
  - Used to make OAuth calls after user is logged in.

Authentication Steps

Logging in to PLM
Pre-step – Get PLM

- Get a developer tenant
  - Log a help request with ADN
- Buy licenses
  - Go to www.autodesk.com
- Get a 30 day trial
  - Go to www.autodesk.com

Logging in to PLM
Pre-step – Get Oxygen Consumer Keys

Email api.key.request@autodesk.com with the following information:

- Your Name
- Email address
- Company name
- Autodesk Developer Network member ID
- Application name
- Type of application – web/desktop/mobile
  - If 'web', the URL of application
- URL of PLM 360 tenant
- Brief description of the application

Warning

- Logging in is hard.
- I’m not kidding. It’s very very hard.
- Everything else is easy after you log in.

Logging in to PLM
Step 1 - Getting the Request Token

Make a request to https://accounts.autodesk.com/OAuth/RequestToken

Request must be signed with your Consumer Secret.

Set the callback to “oob” for desktop and mobile apps.

Logging in to PLM
Step 1 - Getting the Request Token

Example request (header):

oauth_callback=oob
oauth_nonce=6f2a7d5d-1ee8-4cfe-8f22-b3ff8e3b4d5e
oauth_version=1.0
oauth_signature_method=HMAC-SHA1
oauth_consumer_key=213f6e53-19fa-4b22-9bfb-5215fbfa7a93
oauth_timestamp=1375903587
oauth_signature=pWEgqm8XQ0MLVOz78sVDwzP96Oo

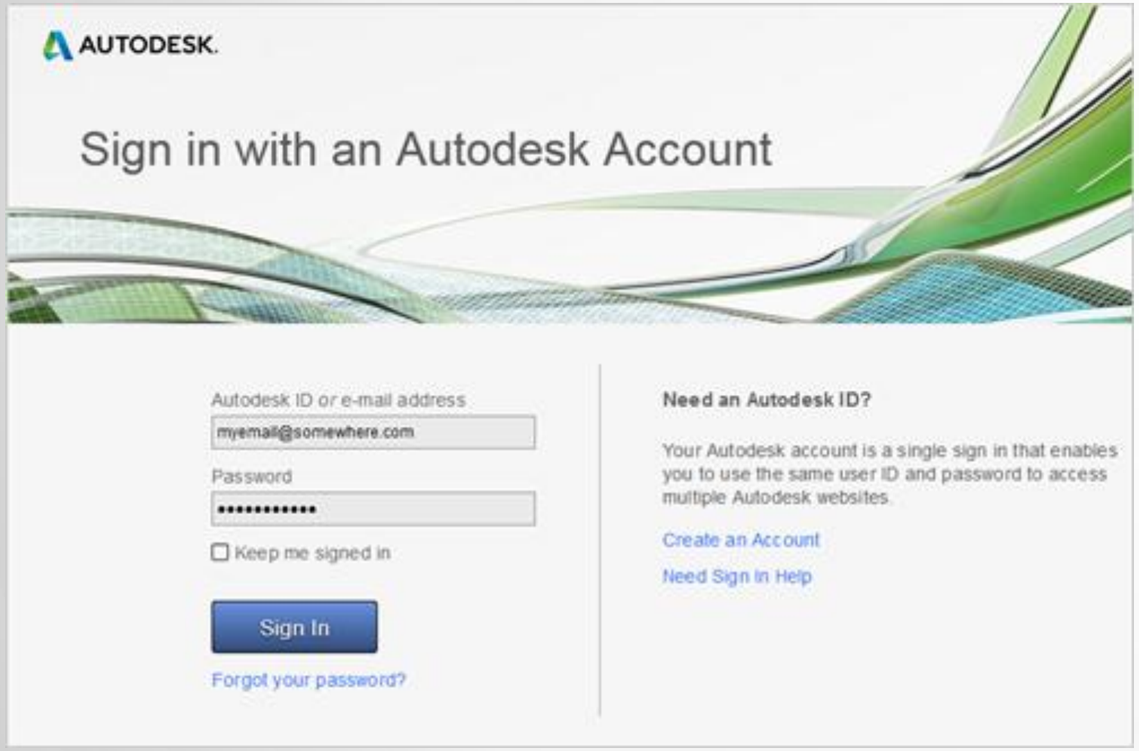
Logging in to PLM
Step 1 - Getting the Request Token

Example response (body):

oauth_token=36%2BdZ74Ig72QxyGtCA8mGJKrCxM%3D&
oauth_token_secret=IY4EAVxObv223m5EOq4YTO7UaH4%3D&
oauth_callback_confirmed=true&
x_oauth_client_identifier=rAXgqmjbb0aqVOzNrsVDwaI96Oo%3D

Logging in to PLM
Step 2 – Launch a Web Browser Control

Send the URL to
https://accounts.autodesk.com/OAuth/Authorize?oauth_token=*requestToken*



For a mobile view, append
&viewmode=mobile
to query string

Logging in to PLM
Step 3 – User Logs In

User types in username/password directly in the web browser.

Your app should not be aware of the user's password.

Your app should not be storing password information.

Your app should not be provide any custom login UI.

Logging in to PLM
Step 4 – Intercepting the Redirect

After a successful login, Oxygen does a redirect.

The redirect will have a **verifier** string.

Example: BHePF8wenS

Your app intercepts the redirect and reads the verifier.

Logging in to PLM
Step 5 – Getting the Access Token

Make a request to https://accounts.autodesk.com/OAuth/AccessToken

Set the verifier (from step 4).

Set the “token” to value of the request token (from step 1)

Request must be signed with your Consumer Secret and Request Secret.

Logging in to PLM
Step 5 – Getting the Access Token

Example request (header):

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

Logging in to PLM
Step 5 – Getting the Access Token

Example response (body):

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

Logging in to PLM
Step 6 –Creating OxygenLoginCredentials

The next request is to PLM 360.

https://*yourTenantName*.autodeskplm360.net/api/v2/authentication/oxygen-login is the REST URL.

It requires an OxygenLoginCredentials object to be passed in.

OxygenLoginCredentials has two parts to it:

- **customerId** – Your tenant name (upper case)
- **validation** – OAuth information

Logging in to PLM
Step 6 –Creating OxygenLoginCredentials

Example OxygenLoginCredentials JSON:


```
{
  "customerId":"MYSITE",
  "validation":"OAuth
  oauth_signature\u003d\"07q79mKOGWgnqri5ydfw2tUndLI%3D\",
  oauth_version\u003d\"1.0\",
  oauth_nonce\u003d\"88125ebd-5417-4df2-ba47-c045b38d575c\",
  oauth_consumer_key\u003d\"213f6e53-19fa-4b22-9bfb-5215fbfa7a93\",
  oauth_signature_method\u003d\"HMAC-SHA1\",
  oauth_timestamp\u003d\"1375904210\",
  oauth_token\u003d\"KLe2eC3792uDhz1rnznFVjr9ibI%3D\""
}
```


Validation is same as header for an OAuth request

Logging in to PLM
Step 6 –Creating OxygenLoginCredentials

Validation data:

- **requestUrl** – full URL to oxygen-login
- **requestMethod** – use “POST” as the value
- **oauth_consumer_key**
- **oauth_signature_method**
- **oauth_version**
- **oauth_timestamp**
- **oauth_nonce**
- **oauth_token** – use the public Access Token
- **oauth_signature** – use Consumer Secret and Access Secret

Logging in to PLM
Step 6 –Creating OxygenLoginCredentials

Two ways to get the validation string:

- Create it by hand (C# example)
- Create and sign an OAuth request but don’t send. Read header off the request object.

Logging in to PLM
Step 7 – Sending OxygenLoginCredentials

Call the “oxygen-login” URL.

If all goes well, you will be back a Session object.
Example response JSON:

```
{
  "userId" : "JohnDoe",
  "customerToken" : "MYSITE"
}
```

Logging in to PLM
Step 7 – Sending OxygenLoginCredentials

Save cookies from response.

Use the cookies in future REST v2 requests.

From this point on:

- No more OAuth
- No more signing

API Features

API Features

REST URL hierarchy matches PLM hierarchy:

- Workspaces
  - Items
    - Files

API Features

Example GET calls:

/api/v2/workspaces
/api/v2/workspaces/8
/api/v2/workspaces/8/items
/api/v2/workspaces/8/items/176
/api/v2/workspaces/8/items/176/files
/api/v2/workspaces/8/items/176/files/287

API Features – Workspace features

- Get a flat list of workspaces
- Get the list of workspaces as shown in the menu
- Get basic workspace information (name, type)
- Get Item Details fields
- No support for add/update/delete

API Features – Item features

- Get flat list of items in a workspace
- Get filtered list of items in a workspace
- Get item detail values
- Add / delete items
- Update item detail values

Supported special cases:

- Binary fields (images)
- Picklist fields

API Features - Files

- Get meta data (file name, size)
- Download (latest only)
- Add / delete
- Checkout / checkin
- Undo checkout

API Features - Paging

- If an API call returns a list, it is always a PagedCollection<T>.
- Not all endpoints support paging.
- Paging endpoints allow “page” and “page-size” query strings.

API Features – Data Types

- Integer
- Decimal
- Date
- DateTime
- String
- Boolean
- BLOB
- Picklist

API Features - null

- Everything is nullable.
- Null is different than “not in JSON”.
  - Setting an item field to null will clear the values.
  - Omitting an item field will leave the existing value alone.
- Null and empty array are the same.

API Features - Security

REST data conforms to permissions of logged in user.

- If user doesn’t have read access on object result is a NOT_FOUND error.
- Item fields are omitted if user doesn’t have read access to the field.
- Data within a payload may be nulled if user doesn’t have read access to the data.

REST Features - Enumerations

- Sent as string over JSON.
- Enumerations may expand over time.
- Best practice – deserialize to a string.
- UNSUPPORTED value in all PLM API enums.
  - Best practice – deserialize unknown values to “UNSUPPORTED”

REST Features - Errors

- Errors will return an HTTP code along with JSON in the body.
- JSON provides PLM-specific error information.
- Data includes:
  - PLM error code
  - Human readable message (English). Intended for error log and developer.

Wrapup

More Information

- PLM 360 and Vault Development Blog
  http://justonesandzeros.typepad.com/
- PLM 360 Online Help
  http://help.autodesk.com/view/PLM/ENU/
- PLM 360 Discussion Group
  http://forums.autodesk.com/t5/PLM-360-General/bd-p/705

Survey Reminder

- Please fill out the survey.