AUTODESK **UNIVERSITY** 2015

SD10623

# Customizing BIM 360 Field with Your Own Apps

Jaime Rosales Duque
Autodesk, Inc.

## Learning Objectives

- Learn how to start development with BIM 360 Field API
- Learn how to create workflows using REST apps for a faster development
- Discover BIM 360 Field API-supported scenarios/workflows
- Learn useful resources for the BIM 360 Field API

## Description

This class will cover how to customize BIM 360 Field software to work with your own apps using its API (Application Programming Interface). We will also cover best practices on how to use REST client applications to work faster and get the most out of the BIM 360 Field API. We will show 4 C# projects to demonstrate the BIM 360 Field API and show how you can get started faster. Finally, the class will discuss the current state of the API and the various scenarios and workflows supported. Please note that prior programming knowledge is required (Microsoft .NET) and an understanding of REST web services and the JavaScript API is recommended. This class is not suitable for developers at the beginner level.

## Your AU Experts

***Jaime Rosales*** *is a Senior Developer Consultant at Autodesk Developer Network (ADN) and has been a member of DevTech Americas Team since July 2014. Jaime joined Autodesk, Inc., in 2011 through the acquisition of Horizontal Systems, the company that developed the cloud-based collaboration systems now known as BIM 360 Glue software (the Glue). He was responsible for developing all the add-ins for BIM 360 Glue, using the APIs of various desktop products, such as Revit software, Navisworks software, and AutoCAD software.*

*Jaime was also a part of the data integration work that connects BIM 360 Glue, BIM 360 Field software, and Navisworks software in a real-time collaboration workflow, using the 3 products' APIs.*

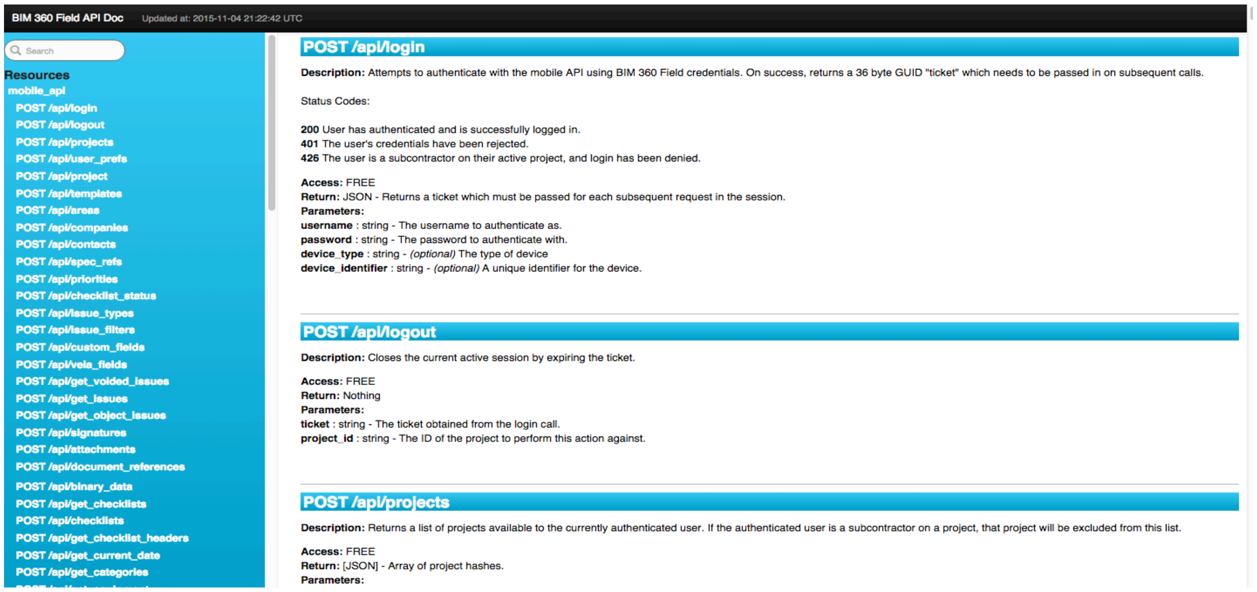
*He is based in New York City, where he has joined the community of JavaScript API developers at different meetups, giving him the chance to present the new View and Data API that A360 cloud-computing platform uses. Jaime also attends many architecture, engineering, and construction (AEC) Hackathons all over the world evangelizing on AEC industry technologies by Autodesk.*

## Learn how to start development with BIM 360 Field API

### Field API Architecture

As with any other product, the Field APIs have also evolved over time. Historically, these APIs originated from the need to interface with the mobile app and has grown to support the common integration requirements. The long-term vision is to have them unified and similar to the BIM 360 product family. The documentation lives in the Field API webpage (figure 1).

**https://bim360field.autodesk.com/apidoc/index.html**



BIM 360 Field API Doc Updated at: 2015-11-04 21:22:42 UTC

Resources
mobile_api
POST /api/login
POST /api/logout
POST /api/projects
POST /api/user_prefs
POST /api/project
POST /api/templates
POST /api/areas
POST /api/companies
POST /api/contacts
POST /api/spec_refs
POST /api/priorities
POST /api/checklist_status
POST /api/issue_types
POST /api/issue_filters
POST /api/custom_fields
POST /api/vela_fields
POST /api/get_voided_issues
POST /api/get_issues
POST /api/get_object_issues
POST /api/signatures
POST /api/attachments
POST /api/document_references
POST /api/binary_data
POST /api/get_checklists
POST /api/checklists
POST /api/get_checklist_headers
POST /api/get_current_date
POST /api/get_categories

**POST /api/login**

**Description:** Attempts to authenticate with the mobile API using BIM 360 Field credentials. On success, returns a 36 byte GUID "ticket" which needs to be passed in on subsequent calls.

Status Codes:

**200** User has authenticated and is successfully logged in.
**401** The user's credentials have been rejected.
**426** The user is a subcontractor on their active project, and login has been denied.

**Access:** FREE
**Return:** JSON - Returns a ticket which must be passed for each subsequent request in the session.
**Parameters:**
**username** : string - The username to authenticate as.
**password** : string - The password to authenticate with.
**device_type** : string - *(optional)* The type of device
**device_identifier** : string - *(optional)* A unique identifier for the device.

**POST /api/logout**

**Description:** Closes the current active session by expiring the ticket.

**Access:** FREE
**Return:** Nothing
**Parameters:**
**ticket** : string - The ticket obtained from the login call.
**project_id** : string - The ID of the project to perform this action against.

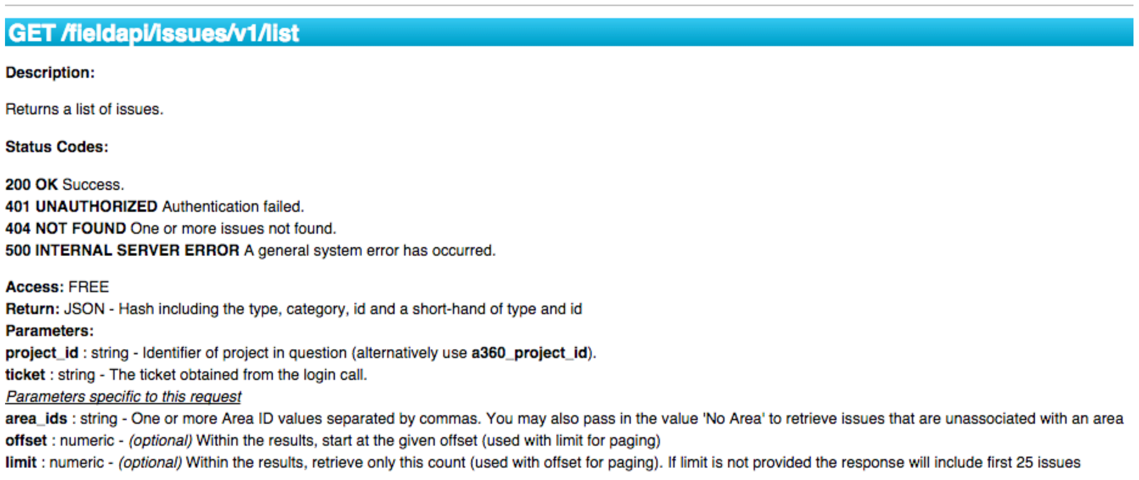
**POST /api/projects**

**Description:** Returns a list of projects available to the currently authenticated user. If the authenticated user is a subcontractor on a project, that project will be excluded from this list.

**Access:** FREE
**Return:** [JSON] - Array of project hashes.
**Parameters:**

*Figure 1 – Field API Documentation*

The end points are in the following structure '**/fieldapi/{object}/v{version}**' (Figure 2) where object corresponds to the high level class of object such as issues or library, etc. and the end points are also versioned. Please note: the results are returned only in JSON, there is no option for receiving responses in XML format.



**GET /fieldapi/issues/v1/list**

**Description:**

Returns a list of issues.

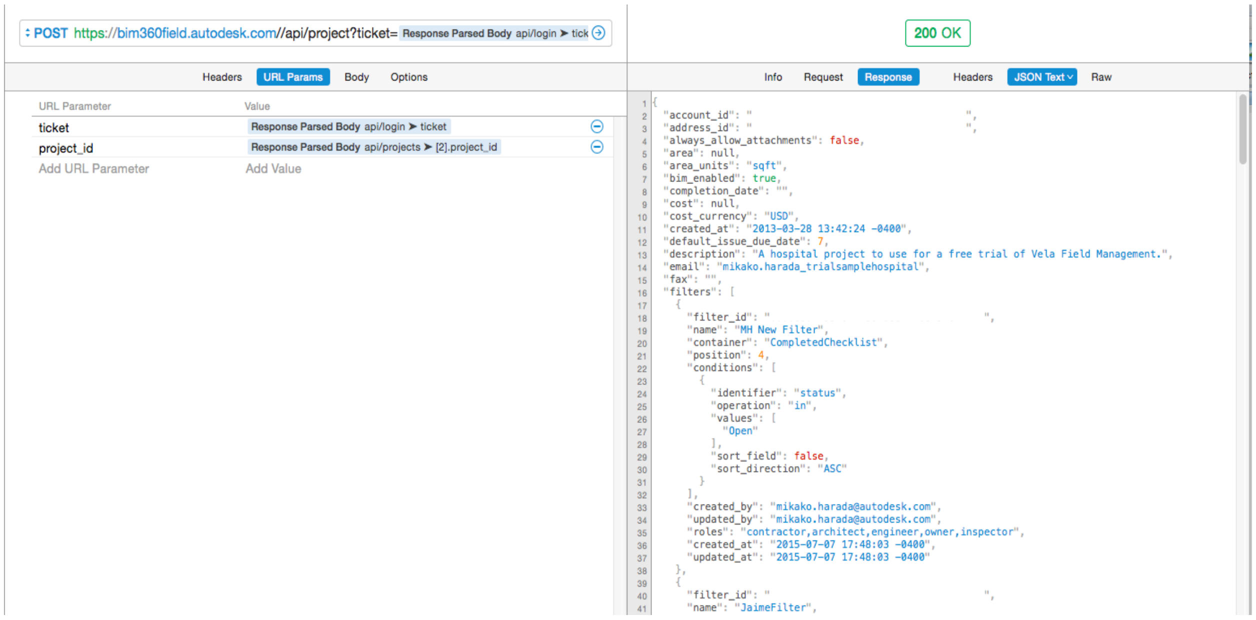
**Status Codes:**

**200 OK** Success.
**401 UNAUTHORIZED** Authentication failed.
**404 NOT FOUND** One or more issues not found.
**500 INTERNAL SERVER ERROR** A general system error has occurred.

**Access:** FREE
**Return:** JSON - Hash including the type, category, id and a short-hand of type and id
**Parameters:**
**project_id** : string - Identifier of project in question (alternatively use **a360_project_id**).
**ticket** : string - The ticket obtained from the login call.
*Parameters specific to this request*
**area_ids** : string - One or more Area ID values separated by commas. You may also pass in the value 'No Area' to retrieve issues that are unassociated with an area
**offset** : numeric - *(optional)* Within the results, start at the given offset (used with limit for paging)
**limit** : numeric - *(optional)* Within the results, retrieve only this count (used with offset for paging). If limit is not provided the response will include first 25 issues

*Figure 2 – Field API Call Structure*

The Field web services are REST based that has specific end points that support CRUD operations and returns results serialized as JSON. (figure 3)



POST https://bim360field.autodesk.com//api/project?ticket= Response Parsed Body api/login ➤ tick

200 OK

| URL Parameter | Value |
|---|---|
| ticket | Response Parsed Body api/login ➤ ticket |
| project_id | Response Parsed Body api/projects ➤ [2].project_id |
| Add URL Parameter | Add Value |

```
{
  "account_id": "                              ",
  "address_id": "                              ",
  "always_allow_attachments": false,
  "area": null,
  "area_units": "sqft",
  "bim_enabled": true,
  "completion_date": "",
  "cost": null,
  "cost_currency": "USD",
  "created_at": "2013-03-28 13:42:24 -0400",
  "default_issue_due_date": 7,
  "description": "A hospital project to use for a free trial of Vela Field Management.",
  "email": "mikako.harada_trialsamplehospital",
  "fax": "",
  "filters": [
    {
      "filter_id": "                              ",
      "name": "MH New Filter",
      "container": "CompletedChecklist",
      "position": 4,
      "conditions": [
        {
          "identifier": "status",
          "operation": "in",
          "values": [
            "Open"
          ],
          "sort_field": false,
          "sort_direction": "ASC"
        }
      ],
      "created_by": "mikako.harada@autodesk.com",
      "updated_by": "mikako.harada@autodesk.com",
      "roles": "contractor,architect,engineer,owner,inspector",
      "created_at": "2015-07-07 17:48:03 -0400",
      "updated_at": "2015-07-07 17:48:03 -0400"
    },
    {
      "filter_id": "                              ",
      "name": "JaimeFilter",
```

*Figure 3 – JSON result*

Here is a snapshot of the **Issues Data Model** (figure 4). The major objects typically contain many minor objects that are also accessible through the API. For example, the Issue object includes minor objects such as Companies, Location, Attachments, etc.

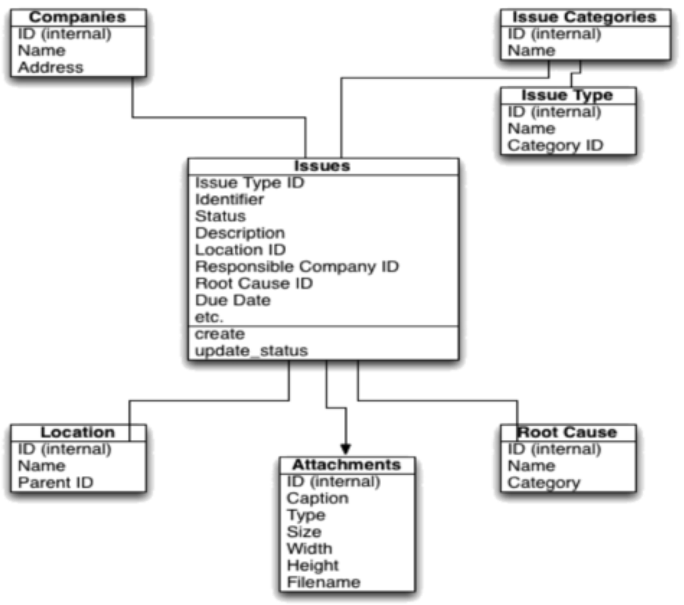


*Figure 4 Issue Data Model*

## Learn how to create workflows using REST apps for a faster development

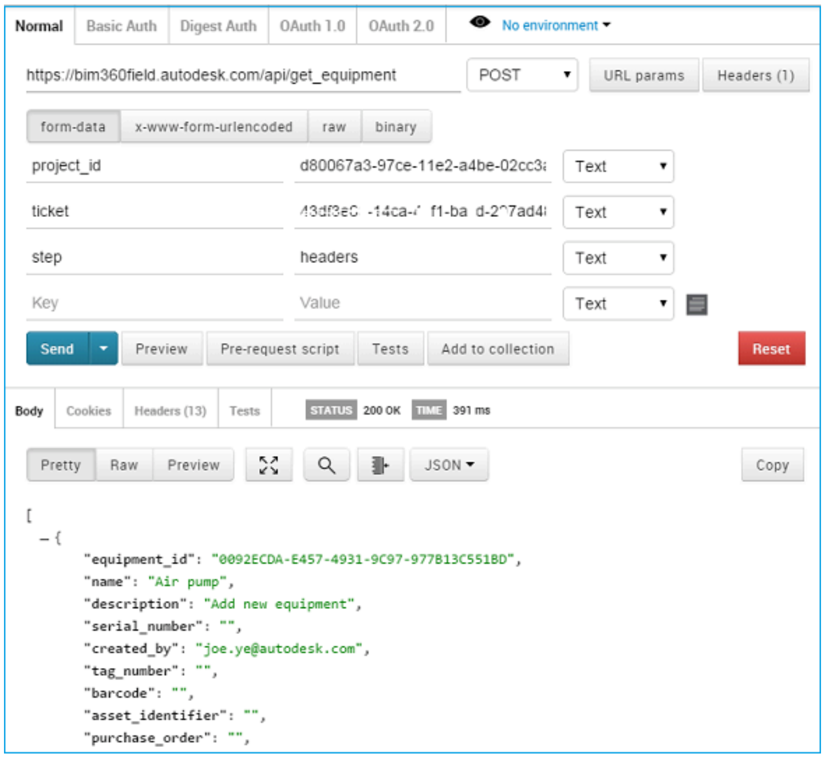
Since we know that Field API it is REST based, I will share with you which tools I have been using to make my development more robust and less time consuming so results are reach faster. As I think it's not just for me and anybody who is using REST API would benefit, I'm going to give a quick summary here. The ones that I'm currently using actively are:

Postman: https://www.getpostman.com/

Paw: https://luckymarmot.com/paw

Online JSON Viewers

**Postman:**
Is a HTTP client tool for testing web services calls, It has a very intuitive UI and anybody who has a basic understanding of REST API can instantly make use of it to put together necessary REST request parameters and test them. It also keeps track of what you have already typed, and allows making a collection to reuse. You can download from chrome web store. You can find a list of features including recording and documentation from the Postman site. There is also a good review on programmable web site, which summarizes the feature nicely. Below shows a sample image from Postman UI.



https://bim360field.autodesk.com/api/get_equipment   POST

| | | |
|---|---|---|
| project_id | d80067a3-97ce-11e2-a4be-02cc3ε | Text |
| ticket | 43df3eC -14ca-' f1-ba d-2^7ad4ε | Text |
| step | headers | Text |
| Key | Value | Text |

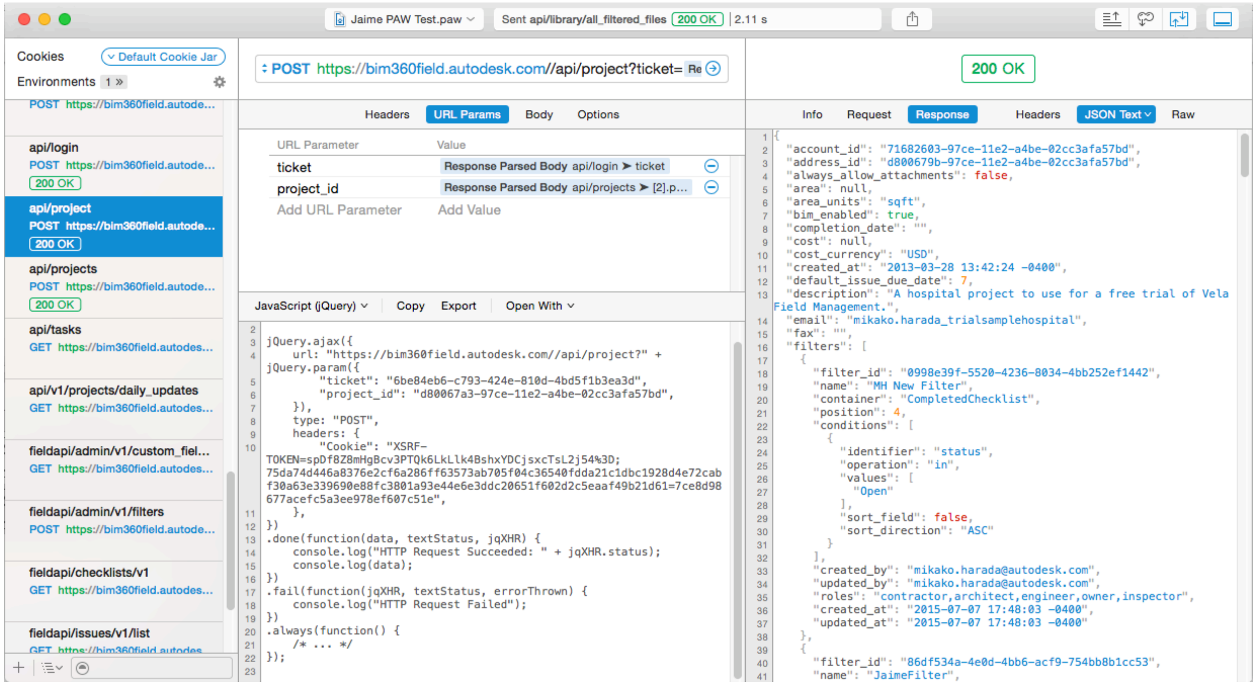
STATUS 200 OK TIME 391 ms

```
[
  – {
        "equipment_id": "0092ECDA-E457-4931-9C97-977B13C551BD",
        "name": "Air pump",
        "description": "Add new equipment",
        "serial_number": "",
        "created_by": "joe.ye@autodesk.com",
        "tag_number": "",
        "barcode": "",
        "asset_identifier": "",
        "purchase_order": "",
```

*A sample image from Postman UI. You can simply specify URL, method and parameters to make a REST call. A response can be formatted for easy viewing.*

## PAW:

Paw is a full-featured designed Mac app that makes interaction with REST services much easier. Whether you are an API maker or consumer, Paw helps you build HTTP requests, inspect the server's response and even generate client code. Using PAW you can build HTTP Requests that will be ready to copy/paste into your code. They support most of the popular code languages; you can even build your own custom code generator via Extensions. It allows you to Access data from previous responses, so you can send back whatever the server returned. Compute hashes, authentication tokens, signatures, randomize data for testing. Paw comes with a large set of Dynamic Values, and you can write your own in JavaScript via Extensions.



POST https://bim360field.autodesk.com//api/project?ticket=

| URL Parameter | Value |
|---|---|
| ticket | Response Parsed Body api/login ➤ ticket |
| project_id | Response Parsed Body api/projects ➤ [2].p... |
| Add URL Parameter | Add Value |

```
jQuery.ajax({
    url: "https://bim360field.autodesk.com//api/project?" +
jQuery.param({
        "ticket": "6be84eb6-c793-424e-810d-4bd5f1b3ea3d",
        "project_id": "d80067a3-97ce-11e2-a4be-02cc3afa57bd",
    }),
    type: "POST",
    headers: {
        "Cookie": "XSRF-
TOKEN=spDf8Z8mHgBcv3PTQk6LkLlk4BshxYDCjsxcTsL2j54%3D;
75da74d446a8376e2cf6a286ff63573ab705f04c36540fdda21c1dbc1928d4e72cab
f30a63e339690e88fc3801a93e44e6e3ddc20651f602d2c5eaaf49b21d61=7ce8d98
677acefc5a3ee978ef607c51e",
    },
})
.done(function(data, textStatus, jqXHR) {
    console.log("HTTP Request Succeeded: " + jqXHR.status);
    console.log(data);
})
.fail(function(jqXHR, textStatus, errorThrown) {
    console.log("HTTP Request Failed");
})
.always(function() {
    /* ... */
});
```

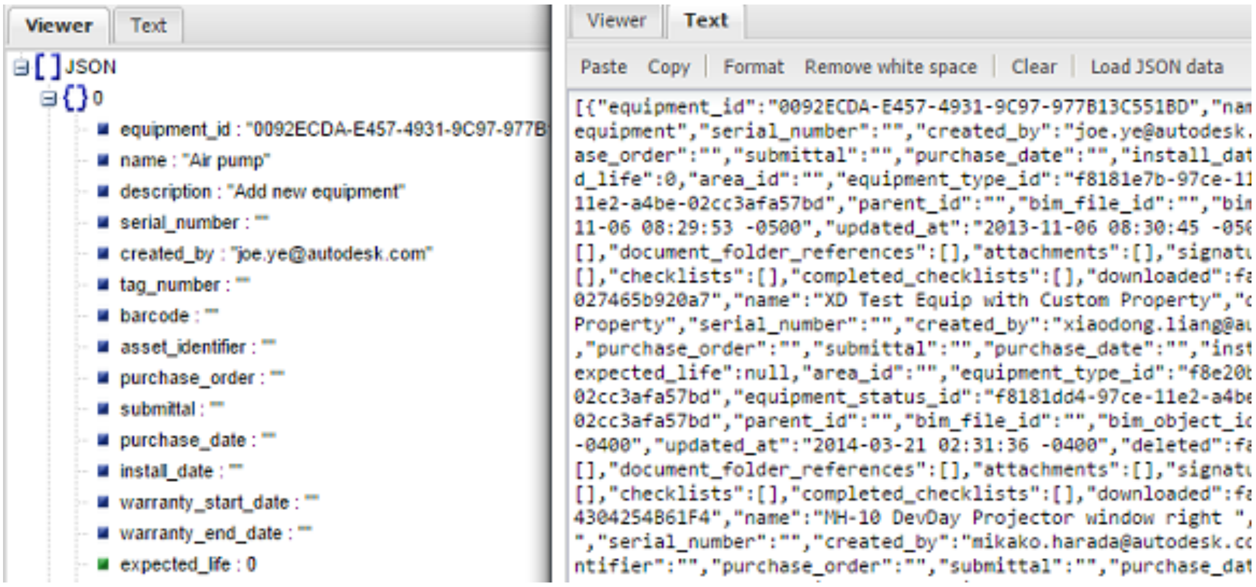
```
{
  "account_id": "71682603-97ce-11e2-a4be-02cc3afa57bd",
  "address_id": "d800679b-97ce-11e2-a4be-02cc3afa57bd",
  "always_allow_attachments": false,
  "area": null,
  "area_units": "sqft",
  "bim_enabled": true,
  "completion_date": "",
  "cost": null,
  "cost_currency": "USD",
  "created_at": "2013-03-28 13:42:24 -0400",
  "default_issue_due_date": 7,
  "description": "A hospital project to use for a free trial of Vela
Field Management.",
  "email": "mikako.harada_trialsamplehospital",
  "fax": "",
  "filters": [
    {
      "filter_id": "0998e39f-5520-4236-8034-4bb252ef1442",
      "name": "MH New Filter",
      "container": "CompletedChecklist",
      "position": 4,
      "conditions": [
        {
          "identifier": "status",
          "operation": "in",
          "values": [
            "Open"
          ],
          "sort_field": false,
          "sort_direction": "ASC"
        }
      ],
      "created_by": "mikako.harada@autodesk.com",
      "updated_by": "mikako.harada@autodesk.com",
      "roles": "contractor,architect,engineer,owner,inspector",
      "created_at": "2015-07-07 17:48:03 -0400",
      "updated_at": "2015-07-07 17:48:03 -0400"
    },
    {
      "filter_id": "86df534a-4e0d-4bb6-acf9-754bb8b1cc53",
      "name": "JaimeFilter",
```

*PAW request with JSON Text Result and JS code generator.*

### Online JSON Viewers

The 3rd one will be a mix of a couple of tools for Online JSON Viewer and Validator that I've been using through the past year when doing Field API development. As the name suggest, is a handy online JSON viewer. While Postman and PAW can handle formatting a REST response directly on their tools, if you simply want to validate a JSON string, this tool gets handy. I use this, for example, when composing a JSON string, and comparing the format of strings between what I see against what a developer sends me to troubleshoot.

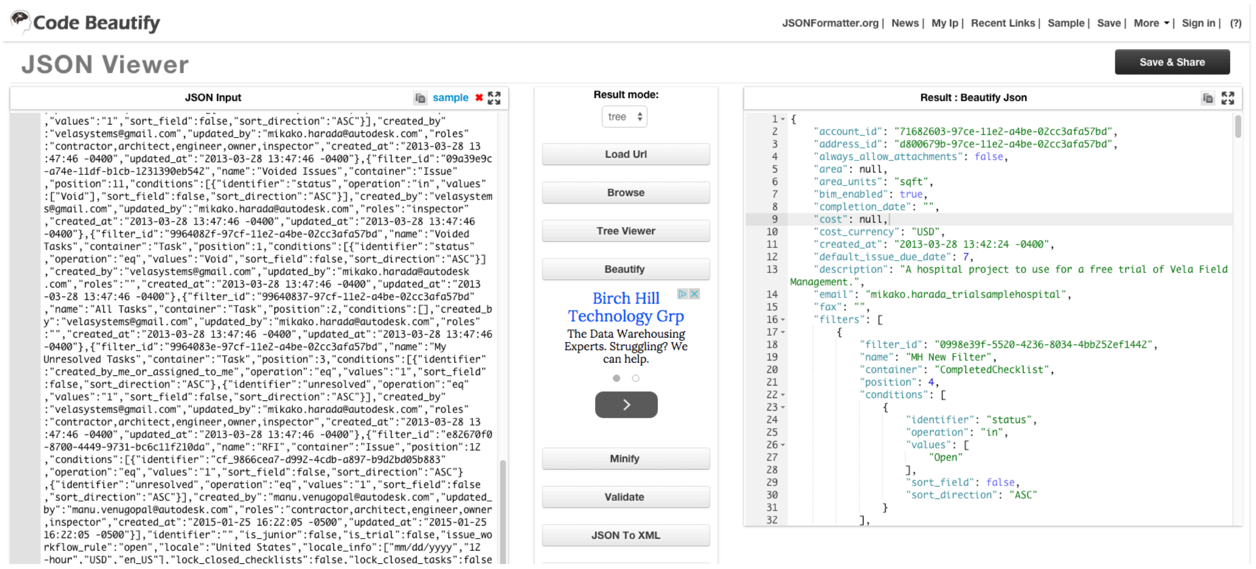
***Online JSON Viewer - http://jsonviewer.stack.hu/***



```
equipment_id : "0092ECDA-E457-4931-9C97-977B
name : "Air pump"
description : "Add new equipment"
serial_number : ""
created_by : "joe.ye@autodesk.com"
tag_number : ""
barcode : ""
asset_identifier : ""
purchase_order : ""
submittal : ""
purchase_date : ""
install_date : ""
warranty_start_date : ""
warranty_end_date : ""
expected_life : 0
```

```
[{"equipment_id":"0092ECDA-E457-4931-9C97-977B13C551BD","nam
equipment","serial_number":"","created_by":"joe.ye@autodesk.
ase_order":"","submittal":"","purchase_date":"","install_dat
d_life":0,"area_id":"","equipment_type_id":"f8181e7b-97ce-11
11e2-a4be-02cc3afa57bd","parent_id":"","bim_file_id":"","bim
11-06 08:29:53 -0500","updated_at":"2013-11-06 08:30:45 -050
[],"document_folder_references":[],"attachments":[],"signatu
[],"checklists":[],"completed_checklists":[],"downloaded":fa
027465b920a7","name":"XD Test Equip with Custom Property","c
Property","serial_number":"","created_by":"xiaodong.liang@au
,"purchase_order":"","submittal":"","purchase_date":"","inst
expected_life":null,"area_id":"","equipment_type_id":"f8e20b
02cc3afa57bd","equipment_status_id":"f8181dd4-97ce-11e2-a4be
02cc3afa57bd","parent_id":"","bim_file_id":"","bim_object_ic
-0400","updated_at":"2014-03-21 02:31:36 -0400","deleted":fa
[],"document_folder_references":[],"attachments":[],"signatu
[],"checklists":[],"completed_checklists":[],"downloaded":fa
4304254B61F4","name":"MH-10 DevDay Projector window right ",
","serial_number":"","created_by":"mikako.harada@autodesk.cc
ntifier":"","purchase_order":"","submittal":"","purchase_dat
```

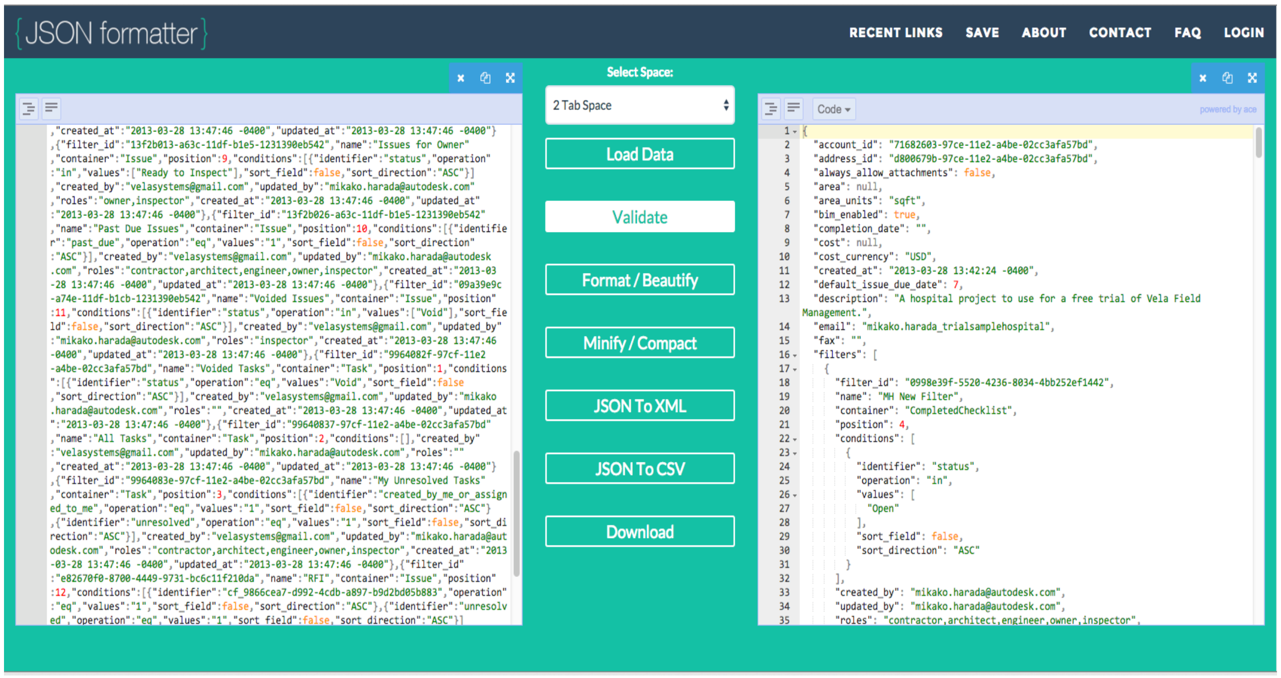
***Code Beautify - http://codebeautify.org/jsonviewer***


Code Beautify
JSONFormatter.org | News | My Ip | Recent Links | Sample | Save | More | Sign in | (?)
JSON Viewer
Save & Share
JSON Input
sample
Result mode:
tree
Load Url
Browse
Tree Viewer
Beautify
Birch Hill Technology Grp
The Data Warehousing Experts. Struggling? We can help.
Minify
Validate
JSON To XML
Result : Beautify Json

```
,"values":"1","sort_field":false,"sort_direction":"ASC"}],"created_by"
:"velasystems@gmail.com","updated_by":"mikako.harada@autodesk.com","roles"
:"contractor,architect,engineer,owner,inspector","created_at":"2013-03-28 13
:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"},{"filter_id":"09a39e9c
-a74e-11df-b1cb-1231390eb542","name":"Voided Issues","container":"Issue"
,"position":11,"conditions":[{"identifier":"status","operation":"in","values"
:["Void"],"sort_field":false,"sort_direction":"ASC"}],"created_by":"velasystem
s@gmail.com","updated_by":"mikako.harada@autodesk.com","roles":"inspector"
,"created_at":"2013-03-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46
-0400"},{"filter_id":"9964082f-97cf-11e2-a4be-02cc3afa57bd","name":"Voided
Tasks","container":"Task","position":1,"conditions":[{"identifier":"status"
,"operation":"eq","values":"Void","sort_field":false,"sort_direction":"ASC"}]
,"created_by":"velasystems@gmail.com","updated_by":"mikako.harada@autodesk
.com","roles":"","created_at":"2013-03-28 13:47:46 -0400","updated_at":"2013
-03-28 13:47:46 -0400"},{"filter_id":"99640837-97cf-11e2-a4be-02cc3afa57bd"
,"name":"All Tasks","container":"Task","position":2,"conditions":[],"created_b
y":"velasystems@gmail.com","updated_by":"mikako.harada@autodesk.com","roles"
:"","created_at":"2013-03-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46
-0400"},{"filter_id":"9964083e-97cf-11e2-a4be-02cc3afa57bd","name":"My
Unresolved Tasks","container":"Task","position":3,"conditions":[{"identifier"
:"created_by_me_or_assigned_to_me","operation":"eq","values":"1","sort_field"
:false,"sort_direction":"ASC"},{"identifier":"unresolved","operation":"eq"
,"values":"1","sort_field":false,"sort_direction":"ASC"}],"created_by"
:"velasystems@gmail.com","updated_by":"mikako.harada@autodesk.com","roles"
:"contractor,architect,engineer,owner,inspector","created_at":"2013-03-28 13
:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"},{"filter_id":"e82670f0
-8700-4449-9731-bc6c11f210da","name":"RFI","container":"Issue","position":12
,"conditions":[{"identifier":"cf_9866cea7-d992-4cdb-a897-b9d2bd05b883"
,"operation":"eq","values":"1","sort_field":false,"sort_direction":"ASC"}
,{"identifier":"unresolved","operation":"eq","values":"1","sort_field":false
,"sort_direction":"ASC"}],"created_by":"manu.venugopal@autodesk.com","updated_
by":"manu.venugopal@autodesk.com","roles":"contractor,architect,engineer,owner
,inspector","created_at":"2015-01-25 16:22:05 -0500","updated_at":"2015-01-25
16:22:05 -0500"}],"identifier":"","is_junior":false,"is_trial":false,"issue_wo
rkflow_rule":"open","locale":"United States","locale_info":["mm/dd/yyyy","12
-hour","USD","en_US"],"lock_closed_checklists":false,"lock_closed_tasks":false
```

```
{
    "account_id": "71682603-97ce-11e2-a4be-02cc3afa57bd",
    "address_id": "d800679b-97ce-11e2-a4be-02cc3afa57bd",
    "always_allow_attachments": false,
    "area": null,
    "area_units": "sqft",
    "bim_enabled": true,
    "completion_date": "",
    "cost": null,
    "cost_currency": "USD",
    "created_at": "2013-03-28 13:42:24 -0400",
    "default_issue_due_date": 7,
    "description": "A hospital project to use for a free trial of Vela Field
Management.",
    "email": "mikako.harada_trialsamplehospital",
    "fax": "",
    "filters": [
        {
            "filter_id": "0998e39f-5520-4236-8034-4bb252ef1442",
            "name": "MH New Filter",
            "container": "CompletedChecklist",
            "position": 4,
            "conditions": [
                {
                    "identifier": "status",
                    "operation": "in",
                    "values": [
                        "Open"
                    ],
                    "sort_field": false,
                    "sort_direction": "ASC"
                }
            ],
```

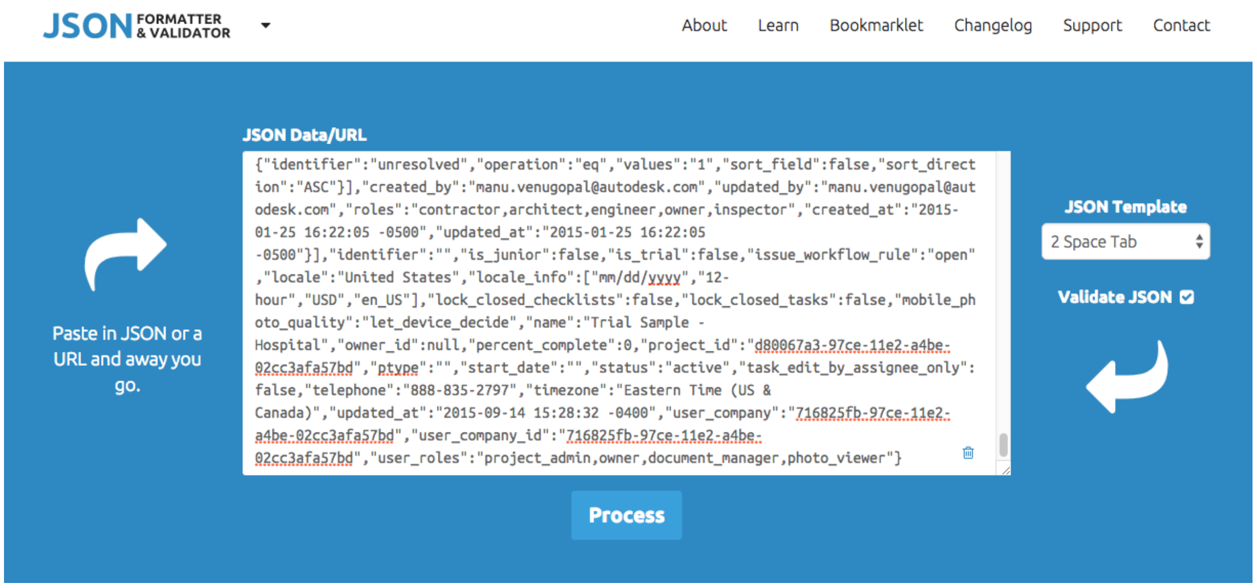
***{JSON Formatter} - http://jsonformatter.org/***



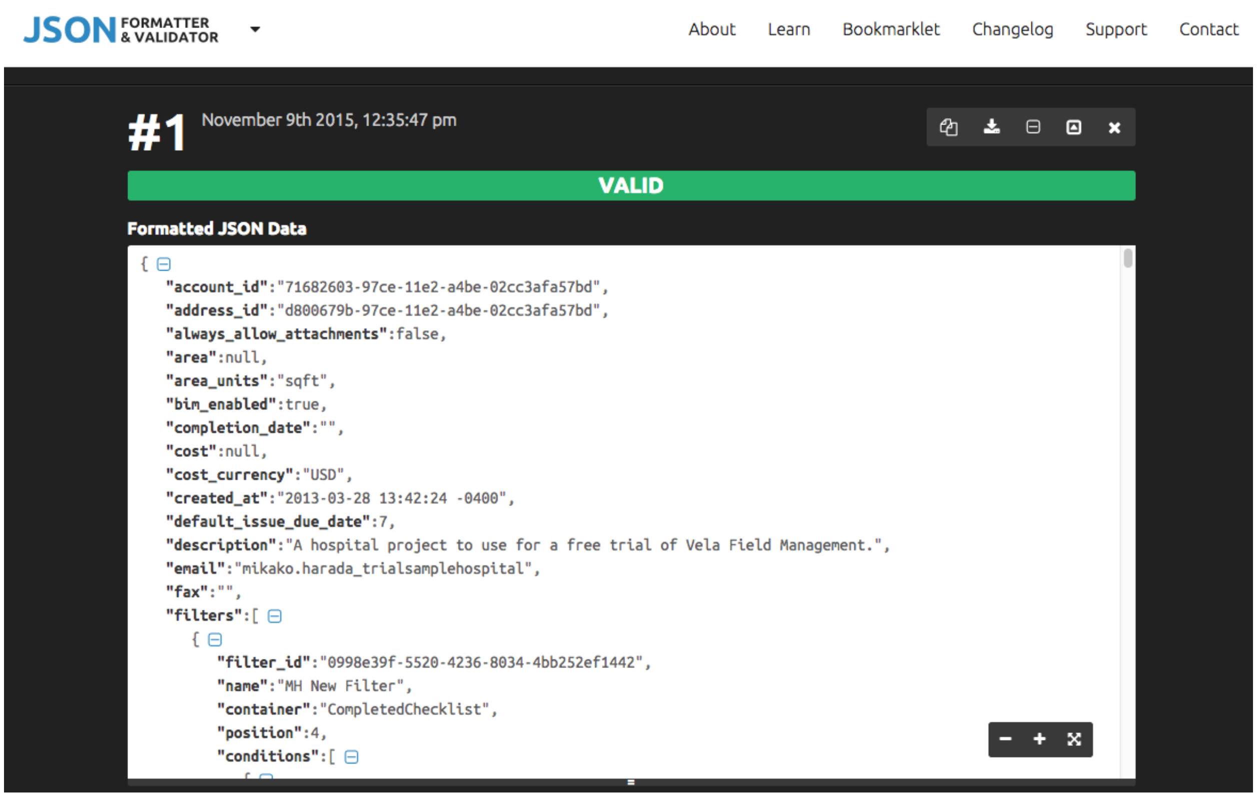
```
,"created_at":"2013-03-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"}
,{"filter_id":"13f2b013-a63c-11df-b1e5-1231390eb542","name":"Issues for Owner"
,"container":"Issue","position":9,"conditions":[{"identifier":"status","operation"
:"in","values":["Ready to Inspect"],"sort_field":false,"sort_direction":"ASC"}]
,"created_by":"velasystems@gmail.com","updated_by":"mikako.harada@autodesk.com"
,"roles":"owner,inspector","created_at":"2013-03-28 13:47:46 -0400","updated_at"
:"2013-03-28 13:47:46 -0400"},{"filter_id":"13f2b026-a63c-11df-b1e5-1231390eb542"
,"name":"Past Due Issues","container":"Issue","position":10,"conditions":[{"identifie
r":"past_due","operation":"eq","values":"1","sort_field":false,"sort_direction"
:"ASC"}],"created_by":"velasystems@gmail.com","updated_by":"mikako.harada@autodesk
.com","roles":"contractor,architect,engineer,owner,inspector","created_at":"2013-03
-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"},{"filter_id":"09a39e9c
-a74e-11df-b1cb-1231390eb542","name":"Voided Issues","container":"Issue","position"
:11,"conditions":[{"identifier":"status","operation":"in","values":["Void"],"sort_fie
ld":false,"sort_direction":"ASC"}],"created_by":"velasystems@gmail.com","updated_by"
:"mikako.harada@autodesk.com","roles":"inspector","created_at":"2013-03-28 13:47:46
-0400","updated_at":"2013-03-28 13:47:46 -0400"},{"filter_id":"9964082f-97cf-11e2
-a4be-02cc3afa57bd","name":"Voided Tasks","container":"Task","position":1,"conditions
":[{"identifier":"status","operation":"eq","values":"Void","sort_field":false
,"sort_direction":"ASC"}],"created_by":"velasystems@gmail.com","updated_by":"mikako
.harada@autodesk.com","roles":"","created_at":"2013-03-28 13:47:46 -0400","updated_at
":"2013-03-28 13:47:46 -0400"},{"filter_id":"99640837-97cf-11e2-a4be-02cc3afa57bd"
,"name":"All Tasks","container":"Task","position":2,"conditions":[],"created_by"
:"velasystems@gmail.com","updated_by":"mikako.harada@autodesk.com","roles":""
,"created_at":"2013-03-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"}
,{"filter_id":"9964083e-97cf-11e2-a4be-02cc3afa57bd","name":"My Unresolved Tasks"
,"container":"Task","position":3,"conditions":[{"identifier":"created_by_me_or_assign
ed_to_me","operation":"eq","values":"1","sort_field":false,"sort_direction":"ASC"}
,{"identifier":"unresolved","operation":"eq","values":"1","sort_field":false,"sort_di
rection":"ASC"}],"created_by":"velasystems@gmail.com","updated_by":"mikako.harada@aut
odesk.com","roles":"contractor,architect,engineer,owner,inspector","created_at":"2013
-03-28 13:47:46 -0400","updated_at":"2013-03-28 13:47:46 -0400"},{"filter_id"
:"e82670f0-8700-4449-9731-bc6c11f210da","name":"RFI","container":"Issue","position"
:12,"conditions":[{"identifier":"cf_9866cea7-d992-4cdb-a897-b9d2bd05b883","operation"
:"eq","values":"1","sort_field":false,"sort_direction":"ASC"},{"identifier":"unresolv
ed","operation":"eq","values":"1","sort_field":false,"sort_direction":"ASC"}]
```

```
{
  "account_id": "71682603-97ce-11e2-a4be-02cc3afa57bd",
  "address_id": "d800679b-97ce-11e2-a4be-02cc3afa57bd",
  "always_allow_attachments": false,
  "area": null,
  "area_units": "sqft",
  "bim_enabled": true,
  "completion_date": "",
  "cost": null,
  "cost_currency": "USD",
  "created_at": "2013-03-28 13:42:24 -0400",
  "default_issue_due_date": 7,
  "description": "A hospital project to use for a free trial of Vela Field
Management.",
  "email": "mikako.harada_trialsamplehospital",
  "fax": "",
  "filters": [
    {
      "filter_id": "0998e39f-5520-4236-8034-4bb252ef1442",
      "name": "MH New Filter",
      "container": "CompletedChecklist",
      "position": 4,
      "conditions": [
        {
          "identifier": "status",
          "operation": "in",
          "values": [
            "Open"
          ],
          "sort_field": false,
          "sort_direction": "ASC"
        }
      ],
      "created_by": "mikako.harada@autodesk.com",
      "updated_by": "mikako.harada@autodesk.com",
      "roles": "contractor,architect,engineer,owner,inspector",
```

***JSON Formatter & Validator - [https://jsonformatter.curiousconcept.com/](https://jsonformatter.curiousconcept.com/)***



```
{"identifier":"unresolved","operation":"eq","values":"1","sort_field":false,"sort_direct
ion":"ASC"}],"created_by":"manu.venugopal@autodesk.com","updated_by":"manu.venugopal@aut
odesk.com","roles":"contractor,architect,engineer,owner,inspector","created_at":"2015-
01-25 16:22:05 -0500","updated_at":"2015-01-25 16:22:05
-0500"}],"identifier":"","is_junior":false,"is_trial":false,"issue_workflow_rule":"open"
,"locale":"United States","locale_info":["mm/dd/yyyy","12-
hour","USD","en_US"],"lock_closed_checklists":false,"lock_closed_tasks":false,"mobile_ph
oto_quality":"let_device_decide","name":"Trial Sample -
Hospital","owner_id":null,"percent_complete":0,"project_id":"d80067a3-97ce-11e2-a4be-
02cc3afa57bd","ptype":"","start_date":"","status":"active","task_edit_by_assignee_only":
false,"telephone":"888-835-2797","timezone":"Eastern Time (US &
Canada)","updated_at":"2015-09-14 15:28:32 -0400","user_company":"716825fb-97ce-11e2-
a4be-02cc3afa57bd","user_company_id":"716825fb-97ce-11e2-a4be-
02cc3afa57bd","user_roles":"project_admin,owner,document_manager,photo_viewer"}
```



```
{
   "account_id":"71682603-97ce-11e2-a4be-02cc3afa57bd",
   "address_id":"d800679b-97ce-11e2-a4be-02cc3afa57bd",
   "always_allow_attachments":false,
   "area":null,
   "area_units":"sqft",
   "bim_enabled":true,
   "completion_date":"",
   "cost":null,
   "cost_currency":"USD",
   "created_at":"2013-03-28 13:42:24 -0400",
   "default_issue_due_date":7,
   "description":"A hospital project to use for a free trial of Vela Field Management.",
   "email":"mikako.harada_trialsamplehospital",
   "fax":"",
   "filters":[
      {
         "filter_id":"0998e39f-5520-4236-8034-4bb252ef1442",
         "name":"MH New Filter",
         "container":"CompletedChecklist",
         "position":4,
         "conditions":[
```

## Discover BIM 360 Field API-supported scenarios/workflows

As a developer you can use a programming environment of your choice (e.g., Microsoft visual studio for C#/.Net, Editor for JavaScript, etc.).

Below are some of the example API service groups and the corresponding end points that are available.

**Admin API** (/fieldapi/admin/v1/) Example - /project_names, /locations, /companies, /users

**Checklists API** (/fieldapi/checklists/v1) Example - /checklists, /checklist (details)

**Issues API** (/fieldapi/issues/v1)

**Equipment, Tasks**, etc.

There is also a set of older generation API methods that follows the structure /**api** instead of the /fieldapi. These are not recommended for use unless the /fieldapi doesn't provide the required functionality. Login method is one such API call that is still provided by /**api**.

Using four Visual Studio Samples we will go over the supported scenarios/workflows we can achieve with the use of the Field API. But before we start reviewing those. We are going to start from the structure on how the information flows through the API.

Let us take a look at how to authenticate against the BIM 360 platform and a common flow of information.

***1. Authenticate using /api/login***

*GET /api/login?username=<insertusername>&password=<insertpassword>*

Response:

```
{
 message: “”,
 title: “”,
 ticket: “cfa003c3-220b-11e3-89a6-02cc3afa57bd”
}
```

***2. Login Result***

The call will return a 36 byte GUID ticket. This must be passed to subsequent calls.

***3. Projects Call***

Make call to /api/projects. This will return an array of project objects that the authenticated user is associated with. Each project object will have a unique project_id value consisting of a 36 byte GUID.

*GET /api/projects?ticket=cfa003c3-220b-11e3-89a6-02cc3afa57bd*

Response:

```
[{
  ...
  project_id: "b24337e0-6937-11e1-af3c-1231390b71c2“
 },
 {...
}]
```

***4. Subsequent calls***

Subsequent calls will require both ticket and project_id. For example, below calls gets the list of checklists in the project specified by the project id.

*GET /fieldapi/checklists/v1?ticket=cfa003c3-220b-11e3-89a6-02cc3afa57bd&project_id=b24337e0-6937-11e1-af3c-1231390b71c2*

Response:

```
[
 {
  status: "Open",
  checklist_type: "QA/QC",
  updated_at: "2012-11-08 09:06:31 -0500",
  created_at: "2012-11-08 09:00:50 -0500",
  name: "Project Closeout List",
  identifier: "000001",
  id: "6CF43FD4-EE35-42FD-84CD-2FEBE5B1208D"
 }
]
```

***5. Checklists***

By specifying the id of the object (in this case the checklist id) you can retrieve the complete information pertaining to the object.

*GET /fieldapi/checklists/v1/6CF43FD4-EE35-42FD-84CD-2FEBE5B1208D?ticket=cfa003c3-220b-11e3-89a6-02cc3afa57bd&project_id=b24337e0-6937-11e1-af3c-1231390b71c2*

Response:

```
{
 {JSON describing a single checklist and its associated objects
 (attachments, items, custom fields, issues, etc)
}
```

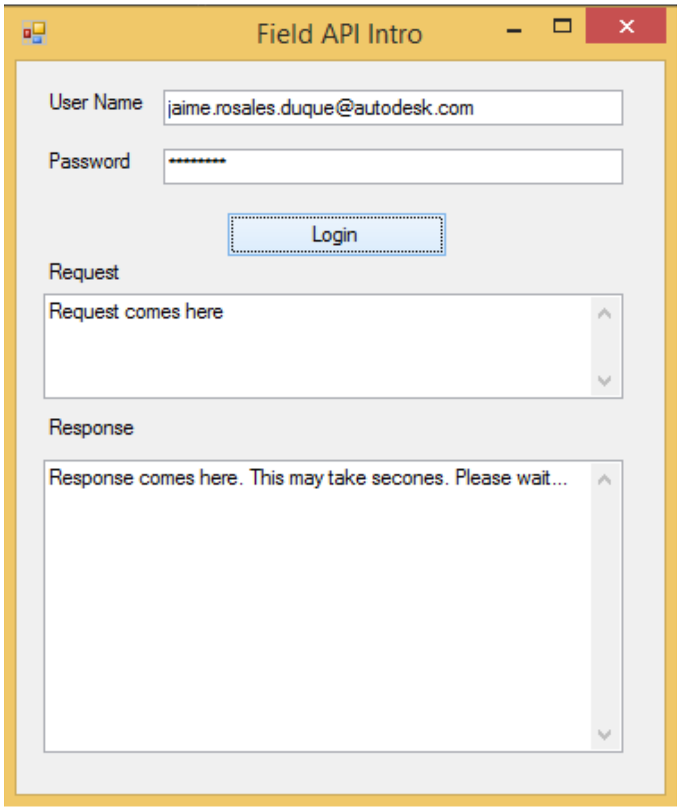
Now that we have a better understanding how the common information flows through the API we can start with the .Net Labs.

https://github.com/BIM360API/FieldAPI_IntroLabs

**Lab 1 – HelloFieldWorld**

In this lab, we learn the first step into the Field API world and the basic of REST API call. This is a simple Windows Forms application written C#. The Field API service we use in this lab is **api/login**, which is a call to sign to the Field and obtain an authentication token or ticket, which is used for subsequent calls.

To make REST calls, we use a library called RestSharp. We chose this as the syntax of RestSharp nicely reflects the semantics of REST call and makes it easy to read the code even if you aren't writing in C#.



User Name: jaime.rosales.duque@autodesk.com

Password: *********

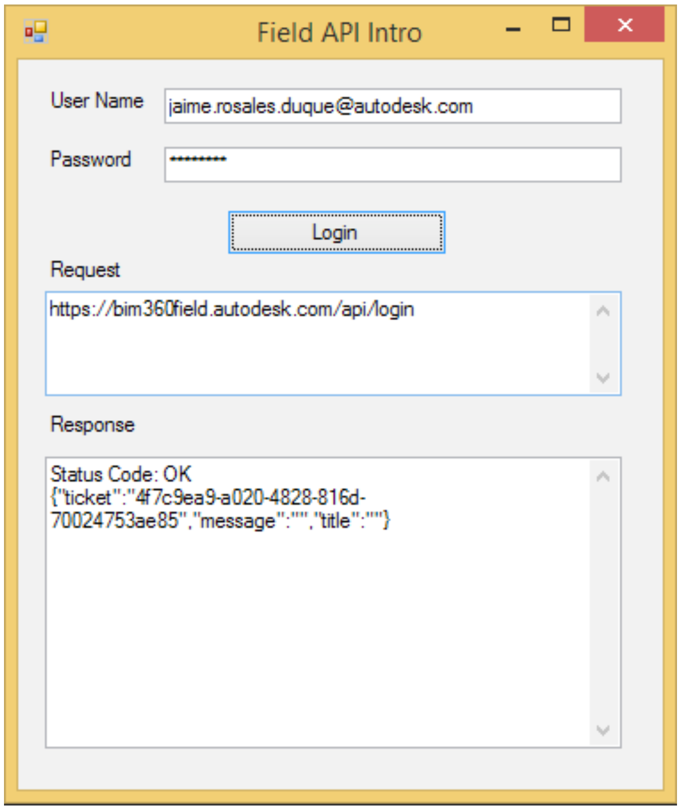
Request comes here

Response comes here. This may take secones. Please wait...

### *Login – Sample Code*

Here is a simple code to show the basic flow of login request. Basically, what we are doing is to (1) fill out the information necessary to send a request, (2) make a web request, and (3) parse the response:

```
public static string Login(string userName, string password)
{
    // (1) Build request
    var client = new RestClient();
    client.BaseUrl = new System.Uri(_baseUrl);

    // Set resource/end point
    var request = new RestRequest();
    request.Resource = "/api/login";
    request.Method = Method.POST;

    // Set required parameters
    request.AddParameter("username", userName);
    request.AddParameter("password", password);

    // (2) Execute request and get response
    IRestResponse response = client.Execute(request);

    // Save response. This is to see the response for our learning.
    m_lastResponse = response;

    // (3) Parse the response and get the ticket.
    string ticket = "";
    if (response.StatusCode == HttpStatusCode.OK)
    {
        JsonDeserializer deserial = new JsonDeserializer();
        LoginResponse loginResponse =
            deserial.Deserialize<LoginResponse>(response);
        ticket = loginResponse.ticket;
    }

    return ticket;
}
```



jaime.rosales.duque@autodesk.com

*********

https://bim360field.autodesk.com/api/login

```
Status Code: OK
{"ticket":"4f7c9ea9-a020-4828-816d-
70024753ae85","message":"","title":""}
```

***Lab 2 – FieldAPIIntro***

In Lab1, we learned how to obtain a ticket or authenticate, and got familiarized ourselves with the basics of REST call. In this lab, we retrieve a list of issues in a project. We then create a new issue.

To achieve this, we use the following services:

/api/login (same as in the Lab1)

/fieldapi/admin/v1/project_names

/fieldapi/issues/v1/list

/fieldapi/issues/v1/create

Work Flow

The workflow of our little app is as follows:

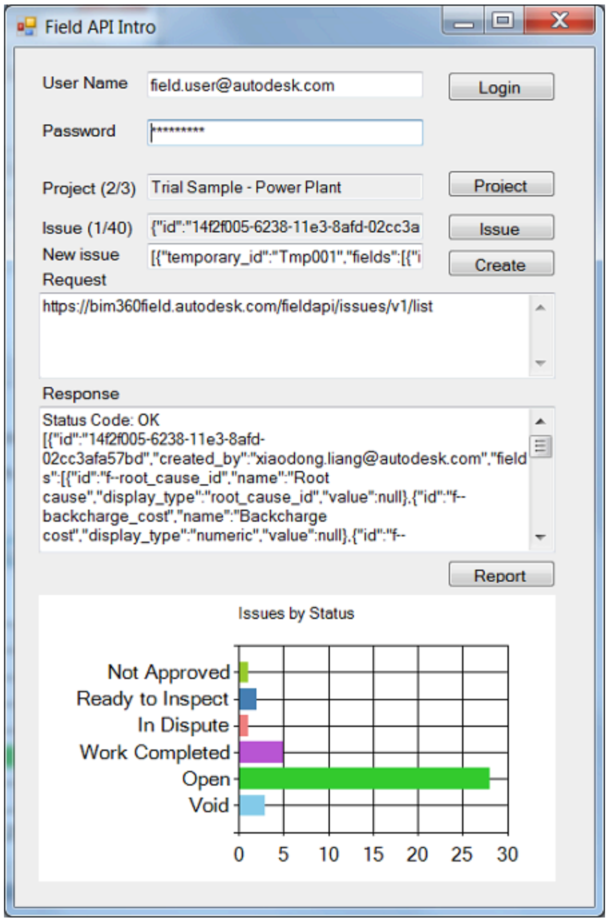
***Login*** – we first log in to the Field and obtains the ticket. We learned how to do this in the Lab1.

***Get a list of projects*** – we obtain a list of currently available projects. We keep one arbitrary project id to further get issues.

***Get a list of issues*** - from the given project, we obtain a list of issues. We take one arbitrary issue from the list, and keep the data for creating a new issue in the next step.

***Create an issue*** – create a new issue, using the issue data obtained in the previous step.

As an example, the application may look something like the following:



```
field.user@autodesk.com
*********
Trial Sample - Power Plant
{"id":"14f2f005-6238-11e3-8afd-02cc3a
[{"temporary_id":"Tmp001","fields":[{"i
https://bim360field.autodesk.com/fieldapi/issues/v1/list
Status Code: OK
[{"id":"14f2f005-6238-11e3-8afd-
02cc3afa57bd","created_by":"xiaodong.liang@autodesk.com","field
s":[{"id":"f--root_cause_id","name":"Root
cause","display_type":"root_cause_id","value":null},{"id":"f-
backcharge_cost","name":"Backcharge
cost","display_type":"numeric","value":null},{"id":"f-
```

Request and Response text boxes are added for our learning purpose.

Note: the chart at the lower portion of the dialog window is not a part of the Field API functionality by itself. We added this as an example. We will take a look at adding a simple chart as a bonus lab in Lab4.

For best purpose of our sample, we are going to use the call project_names instead of the projects call that returns you more information about every individual project.

**Project Names – Sample Code**
Here is the code to obtain the list of Project Names.

```
public static List<Project> ProjectNames(string ticket)
 {
     // (1) Build request
     var client = new RestClient();
     client.BaseUrl = new System.Uri(_baseUrl);

     // Set resource or end point
     var request = new RestRequest();
     request.Resource = "/fieldapi/admin/v1/project_names";
     request.Method = Method.POST;

     // Add parameters
     request.AddParameter("ticket", ticket);
```

```
        // (2) Execute request and get response
        IRestResponse response = client.Execute(request);

        // Save response. This is to see the response for our learning.
        m_lastResponse = response;

        // (3) Parse the response and get the list of projects.

        if (response.StatusCode != HttpStatusCode.OK)
        {
            return null;
        }

        JsonDeserializer deserial = new JsonDeserializer();
        List<Project> proj_list = deserial.Deserialize<List<Project>>(response);

        return proj_list;
    }
```

Where Project is defined as

```
public class Project
{
 public string id { get; set; }
 public string name { get; set; }
 public string a360_project_id { get; set; }
 public string updated_at { get; set; }
}
```

Once you obtained a list of projects (proj_list), you can access project id and its name as follows:

```
Project proj = proj_list[index];
string project_id = proj.id;
string project_name = proj.name;
```

## Issue List – Sample Code

Here is the code sample to obtain a list of issues:

```
 public static List<Issue> IssueList(string ticket, string project_id,
 string area_ids, int offset=0, int limit=20)
 {
     // (1) Build request
     var client = new RestClient();
     client.BaseUrl = new System.Uri(_baseUrl);

     // Set resource or end point
     var request = new RestRequest();
     request.Resource = "/fieldapi/issues/v1/list";
    // /fieldapi/ is prefered over /api/
     request.Method = Method.POST;

     // Add parameters
     request.AddParameter("ticket", ticket);
     request.AddParameter("project_id", project_id);
```

```
        // (2) Execute request and get response
        IRestResponse response = client.Execute(request);

        // Save response. This is to see the response for our learning.
        m_lastResponse = response;

        // (3) Parse the response and get the list of projects.

        if (response.StatusCode != HttpStatusCode.OK)
        {
            return null;
        }

        JsonDeserializer deserial = new JsonDeserializer();
        List<Issue> issue_list = deserial.Deserialize<List<Issue>>(response);

        return issue_list;
    }
```

Where Issue is defined as the code below.

(Note also that we are not using optional parameters in this exercise because of the discrepancy of the result from what the documentation says. We hope it gets clarified near future.)

```
public class Issue
{
        public string id { get; set; }
        public string created_by { get; set; }
        public List<IssueFieldItem> fields { get; set; }
        public List<Object> comments { get; set; }
        public List<Object> attachments { get; set; }
}

public class IssueFieldItem
{
      public string id { get; set; }
      public string name { get; set; }
      public string display_type { get; set; }
      public string value { get; set; }
}
```

**Issue Create – Sample code**

Here is the code sample for Create Issues

```
public static string IssueCreate
(string ticket, string project_id, string issues)
 {
     // (1) Build request
     var client = new RestClient();
     client.BaseUrl = new System.Uri(_baseUrl);

     // Set resource or end point
     var request = new RestRequest();
```

```
            request.Resource = "/fieldapi/issues/v1/create";
            request.Method = Method.POST;

            // Add parameters
            request.AddParameter("ticket", ticket);
            request.AddParameter("project_id", project_id);
            request.AddParameter("issues", issues);

            // (2) Execute request and get response
            IRestResponse response = client.Execute(request);

            // Save response. This is to see the response for our learning.
            m_lastResponse = response;

            // (3) Parse the response and get the list of projects.

            if (response.StatusCode != HttpStatusCode.OK) { returnnull; }

            JsonDeserializer deserial = newJsonDeserializer();
            List<IssueCreateResponse> issueCreateResponse = deserial.Deserialize
           <List<IssueCreateResponse>>(response);
            return issueCreateResponse[0].id;
        }

    }
```

Where IssueCreateResponse is defined as follows:

```
        public class IssueCreateResponse
        {
           public bool success { get; set; }
           public List<object> general_errors { get; set; }
           public List<object> errors { get; set; }
           public string id { get; set; }
           public string temporary_id { get; set; }
        }
```

### Putting Together

Add UI's to call ProjectNames(), IssueNames() and IssueCreate(). This can be of any form. For example, below shows how it looks like for our lab 2 sample application:

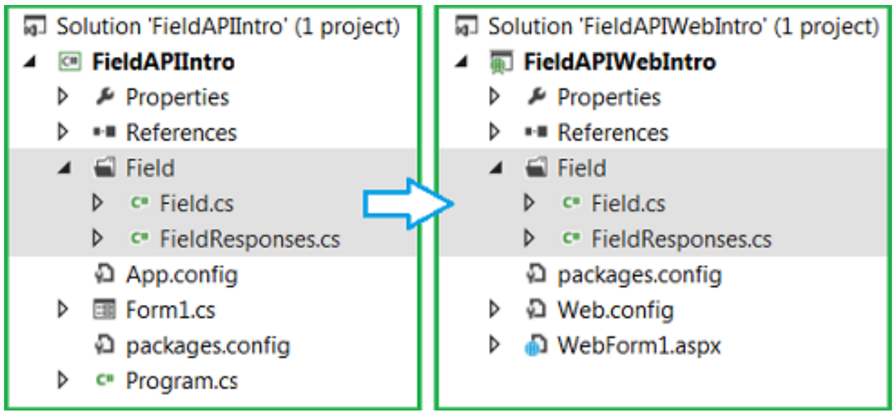
```
       private void buttonProject_Click(object sender, EventArgs e)
        {

            List<Project> proj_list = Field.ProjectNames(m_ticket);

            ShowRequestResponse();

            // Post process to get hold of one project.
            // For simplicity, just pick up arbitrary one for our exercise.

            m_proj_index %= proj_list.Count;
            Project proj = proj_list[m_proj_index++];
            m_project_id = proj.id;
            string project_name = proj.name;
            labelProject.Text = "Project (" + m_proj_index.ToString()
            + "/" + proj_list.Count.ToString() + ")";
            textBoxProject.Text = project_name;
        }

        private void buttonIssue_Click(object sender, EventArgs e)
        {
            string area_ids = "No Area";
            List<Issue> issue_list = Field.IssueList
           (m_ticket, m_project_id, area_ids);
            m_issue_list = issue_list;

            ShowRequestResponse();

            // Post process to get hold of one project.
            // For simplicity, just pick up arbitrary one for our exercise.

            m_issue_index %= issue_list.Count;
            Issue issue= issue_list[m_issue_index++];
            m_issue_id = issue.id;
            IssueFieldItem item = issue.fields.Find(x
            => x.name.Equals("Identifier"));
            labelIssue.Text = "Issue (" + m_issue_index.ToString()
            + "/" + issue_list.Count.ToString() + ")";

            // Show the issue string
            JsonSerializer serial = new JsonSerializer();
            textBoxIssue.Text = serial.Serialize(issue);

            // Make one issue for creation. this is for later use.
            textBoxNewIssue.Text = IssueToString(issue);
        }

        private void buttonIssueCreate_Click(object sender, EventArgs e)
        {
            string issues = textBoxNewIssue.Text;
            string issue_id = Field.IssueCreate(m_ticket, m_project_id, issues);
            ShowRequestResponse();

        }
```

Where IssueToString(issue)takes an issue obtained from IssueList() as an argument and format as an input for IssueCreate(). In our simple app, we define a string for a new issue as a duplicate of the issue we obtained in the previous step. In reality, you will want to implement more elaborate UI to compose a new issue.

Once again, the chart portion is not a part of Field API functionality it was created for educational purposes on what can you do using the API.

### Lab 3 – Field Web API Intro

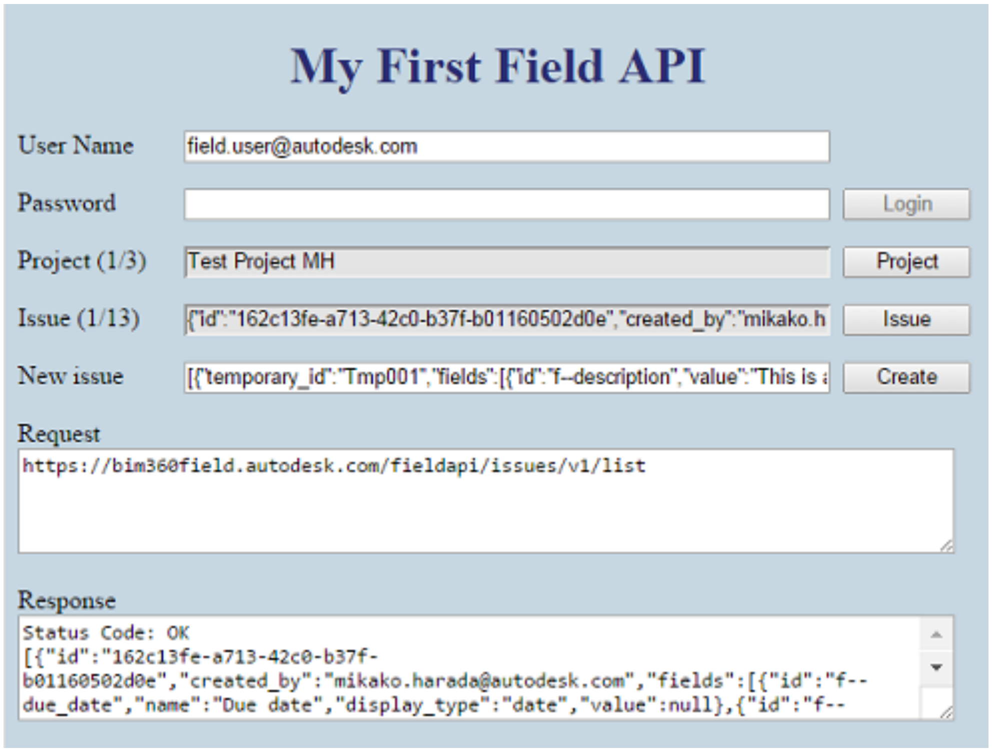
This Lab will follow a very similar structure as the Lab 2, the difference will be that now we will be doing it with a ASP.NET Web Application. The good news is that we have written the functions to make the Field web services REST calls in a way that we can simply reuse them. In the sample project, we put the common Field REST calls under Field folder.



Difference is, of course, the UI is a web page. A web page in ASP.NET has an extension "aspx". ASP.NET tools starts with “asp:” in a web form. A code behind an aspx page has an extension "aspx.cs". In the corresponding code, most of the implementation is analogous to the Win Forms. Only difference is that to save values between different REST calls, we use HttpContext.Current.Session object instead of using a member variable, e.g.

```
// Save the ticket for this session
Session["ticket"] = ticket;

// Retrieve the ticket for this session
string ticket = Session["ticket"] as string;
```

The image below shows you how our Web Form will look now.



My First Field API

User Name: field.user@autodesk.com

Password:

Project (1/3): Test Project MH

Issue (1/13): {"id":"162c13fe-a713-42c0-b37f-b01160502d0e","created_by":"mikako.h

New issue: [{"temporary_id":"Tmp001","fields":[{"id":"f--description","value":"This is a

Request

```
https://bim360field.autodesk.com/fieldapi/issues/v1/list
```

Response

```
Status Code: OK
[{"id":"162c13fe-a713-42c0-b37f-
b01160502d0e","created_by":"mikako.harada@autodesk.com","fields":[{"id":"f--
due_date","name":"Due date","display_type":"date","value":null},{"id":"f--
```

As far as Field API REST calls are concerned, we have covered THE basic of making the calls. The rest will be an application of what we have learned so far.

## Lab 4 – Field API Web Intro Ex

The last piece of functionalities that we want to add to our Intro Labs app is a simple reporting feature. We will add a chart that shows the number of issues sorted by status (i.e., Open, Completed, etc.). Reporting or monitoring the "health" of a project is one of frequently discussed use case scenarios of Field API. As we know, Field is fully REST based, meaning does not have any visualization component, but the data it is available, so we can use it to create charts.

### *Counting Issues by Status*

As data to create a chart, we use the list of issues obtained in the Lab2: i.e., the list returned from `Field.IssueList()`.We simply save it as a session object:

```
        List<Issue> issue_list = Field.IssueList(ticket, project_id);
        Session["issueList"] = issue_list;
```

And retrieve it later when we want to create a chart:

```
        List<Issue> issue_list = Session["issueList"]
        as List<Issue>;
```

Below is the code that counts the number of issues for each status from the given list of issues:

```
// Collect data from the issue list.
// Count the number of issues for each status.
// data is a dictionary of (status, count) pair

protected Dictionary<string, int> CollectData(List<Issue> issue_list)
{
    Dictionary<string, int> data = new Dictionary<string, int>();

    foreach (Issue issue in issue_list)
    {
        string status = issue.fields.Find(x => x.name.Equals("Status")).value;
        if (data.ContainsKey(status))
        {
            data[status]++;
        }
        else
        {
            data.Add(status, 1);
        }
    }

    return data;
}
```

In the above code, this is the line that we needed to know the format of the issue:

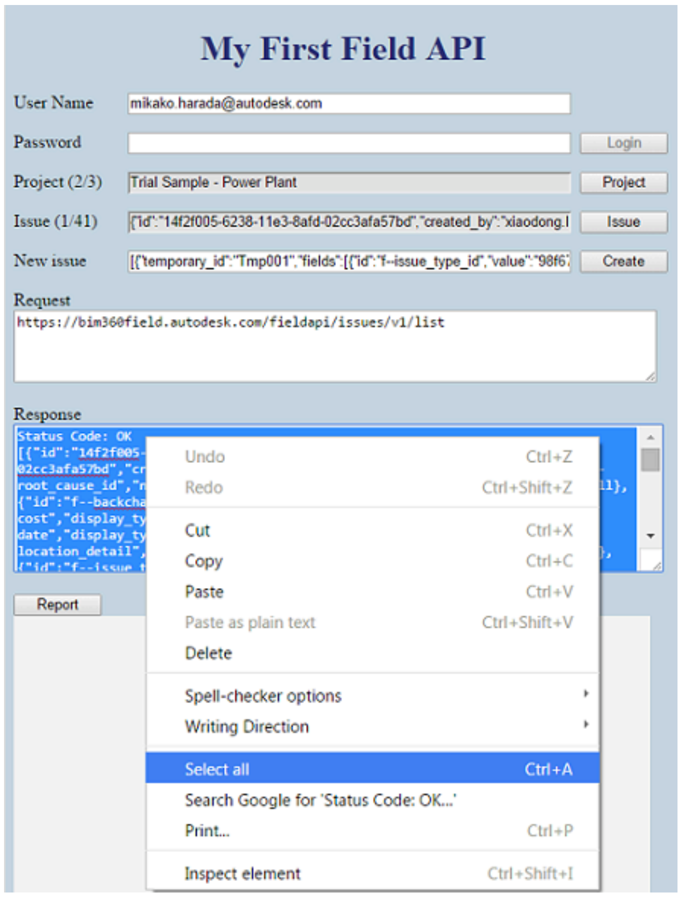
```
string status = issue.fields.Find(x => x.name.Equals("Status")).value;
```

***Re-visiting Issue List***

Here is what I did to decipher the format from an actual response:

1. Using Field API Intro app (either Lab2 or Lab3 will do), make Issues/List call ("Issue" button).

2. In “Response” text box, right click and [Select All] and [Copy]:



My First Field API

| | | |
|---|---|---|
| User Name | mikako.harada@autodesk.com | |
| Password | | Login |
| Project (2/3) | Trial Sample - Power Plant | Project |
| Issue (1/41) | {"id":"14f2f005-6238-11e3-8afd-02cc3afa57bd","created_by":"xiaodong.l | Issue |
| New issue | [{"temporary_id":"Tmp001","fields":[{"id":"f--issue_type_id","value":"98f67 | Create |

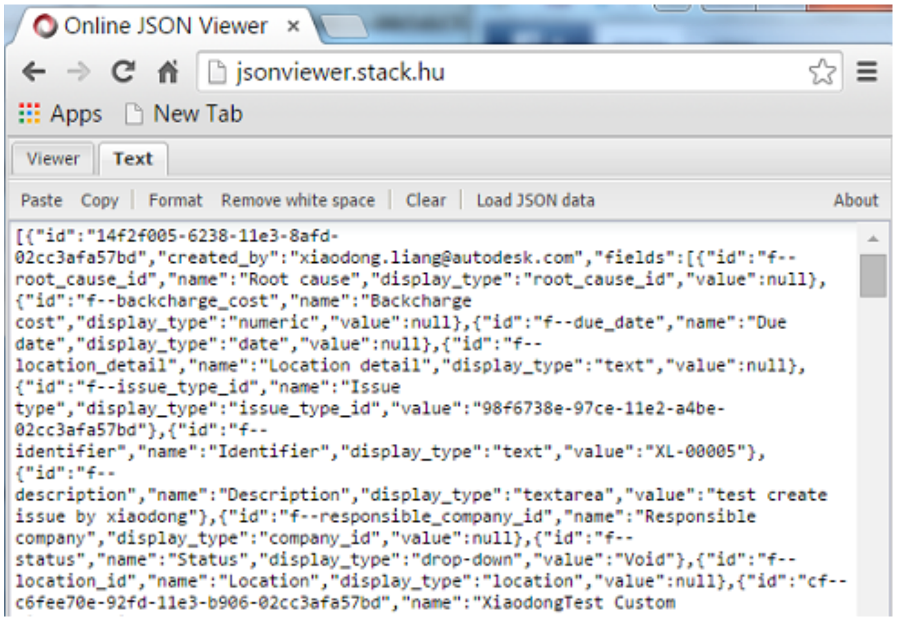
Request

```
https://bim360field.autodesk.com/fieldapi/issues/v1/list
```

Response

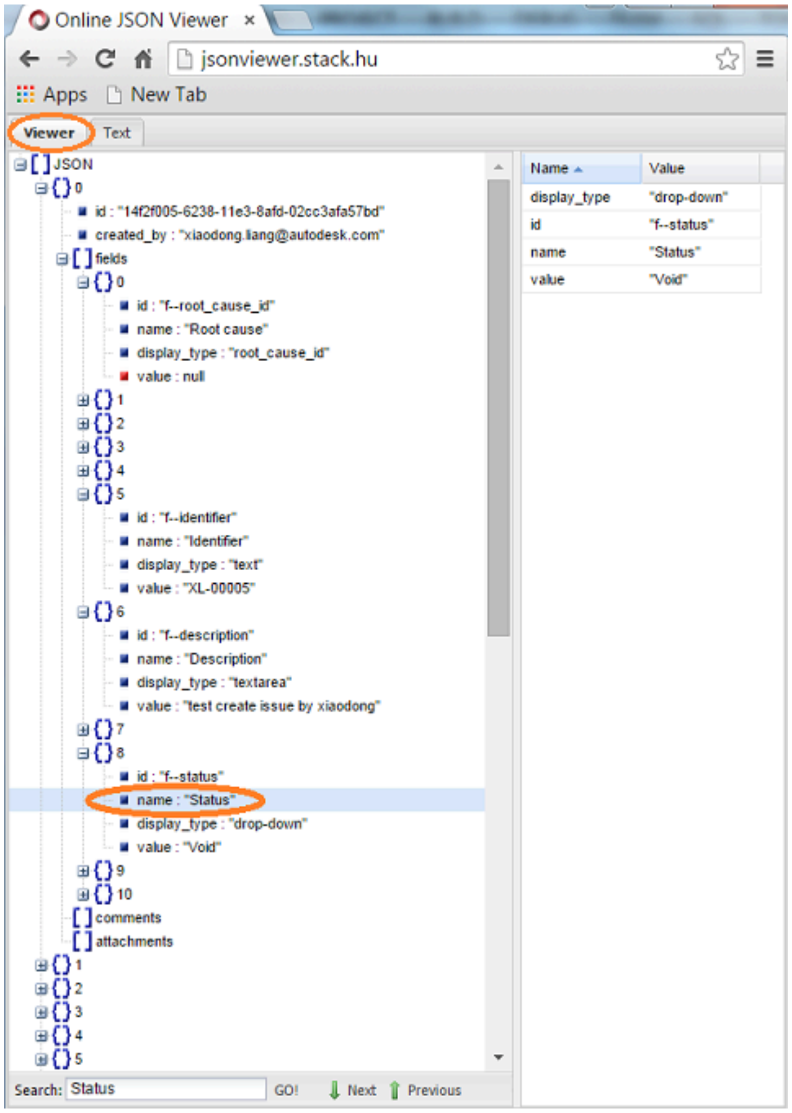
```
Status Code: OK
[{"id":"14f2f005-
02cc3afa57bd","cr
root_cause_id","n
{"id":"f--backcha
cost","display_ty
date","display_ty
location_detail",
{"id":"f--issue_t
```

Now we will use one of the Tools mentioned before to visualize the data of our JSON string.



```
[{"id":"14f2f005-6238-11e3-8afd-
02cc3afa57bd","created_by":"xiaodong.liang@autodesk.com","fields":[{"id":"f--
root_cause_id","name":"Root cause","display_type":"root_cause_id","value":null},
{"id":"f--backcharge_cost","name":"Backcharge
cost","display_type":"numeric","value":null},{"id":"f--due_date","name":"Due
date","display_type":"date","value":null},{"id":"f--
location_detail","name":"Location detail","display_type":"text","value":null},
{"id":"f--issue_type_id","name":"Issue
type","display_type":"issue_type_id","value":"98f6738e-97ce-11e2-a4be-
02cc3afa57bd"},{"id":"f--
identifier","name":"Identifier","display_type":"text","value":"XL-00005"},
{"id":"f--
description","name":"Description","display_type":"textarea","value":"test create
issue by xiaodong"},{"id":"f--responsible_company_id","name":"Responsible
company","display_type":"company_id","value":null},{"id":"f--
status","name":"Status","display_type":"drop-down","value":"Void"},{"id":"f--
location_id","name":"Location","display_type":"location","value":null},{"id":"cf--
c6fee70e-92fd-11e3-b906-02cc3afa57bd","name":"XiaodongTest Custom
```

4. Go to “Viewer” tab, where you can easily examine the response:



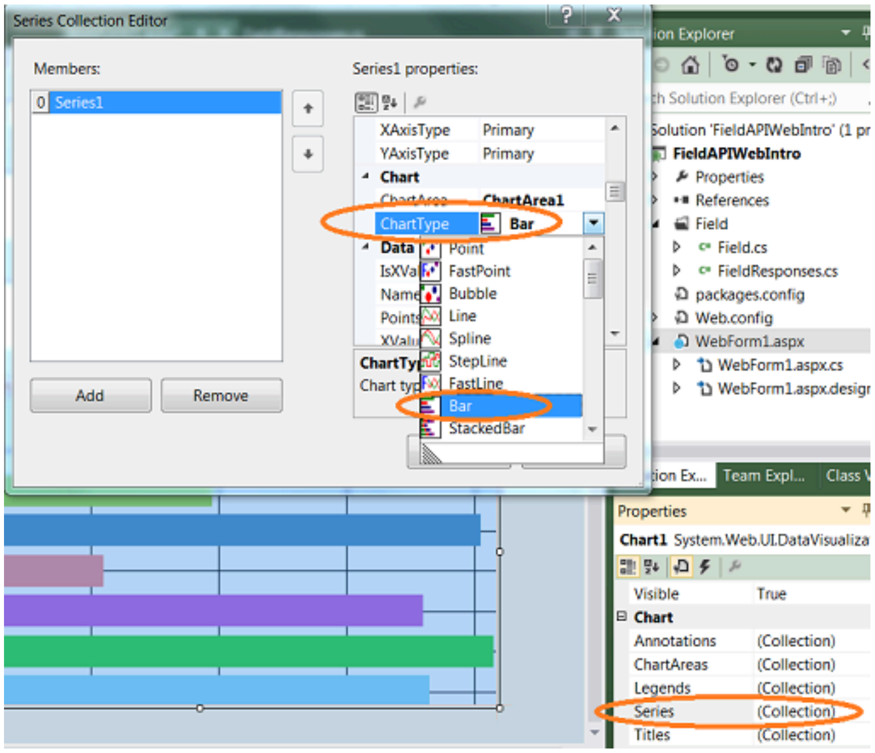
```
[ ] JSON
  { } 0
    id : "14f2f005-6238-11e3-8afd-02cc3afa57bd"
    created_by : "xiaodong.liang@autodesk.com"
    [ ] fields
      { } 0
        id : "f--root_cause_id"
        name : "Root cause"
        display_type : "root_cause_id"
        value : null
      { } 1
      { } 2
      { } 3
      { } 4
      { } 5
        id : "f--identifier"
        name : "Identifier"
        display_type : "text"
        value : "XL-00005"
      { } 6
        id : "f--description"
        name : "Description"
        display_type : "textarea"
        value : "test create issue by xiaodong"
      { } 7
      { } 8
        id : "f--status"
        name : "Status"
        display_type : "drop-down"
        value : "Void"
      { } 9
      { } 10
    [ ] comments
    [ ] attachments
  { } 1
  { } 2
  { } 3
  { } 4
  { } 5
```

| Name | Value |
|---|---|
| display_type | "drop-down" |
| id | "f--status" |
| name | "Status" |
| value | "Void" |

For example, you can see each issue has “fields”, which is a collection of “field”, where each “field” has "id", "name", "display_type" and "value". In our case, we are interested in the value of field whose name is equals to “Status” for each issue.

**Adding a Chart**

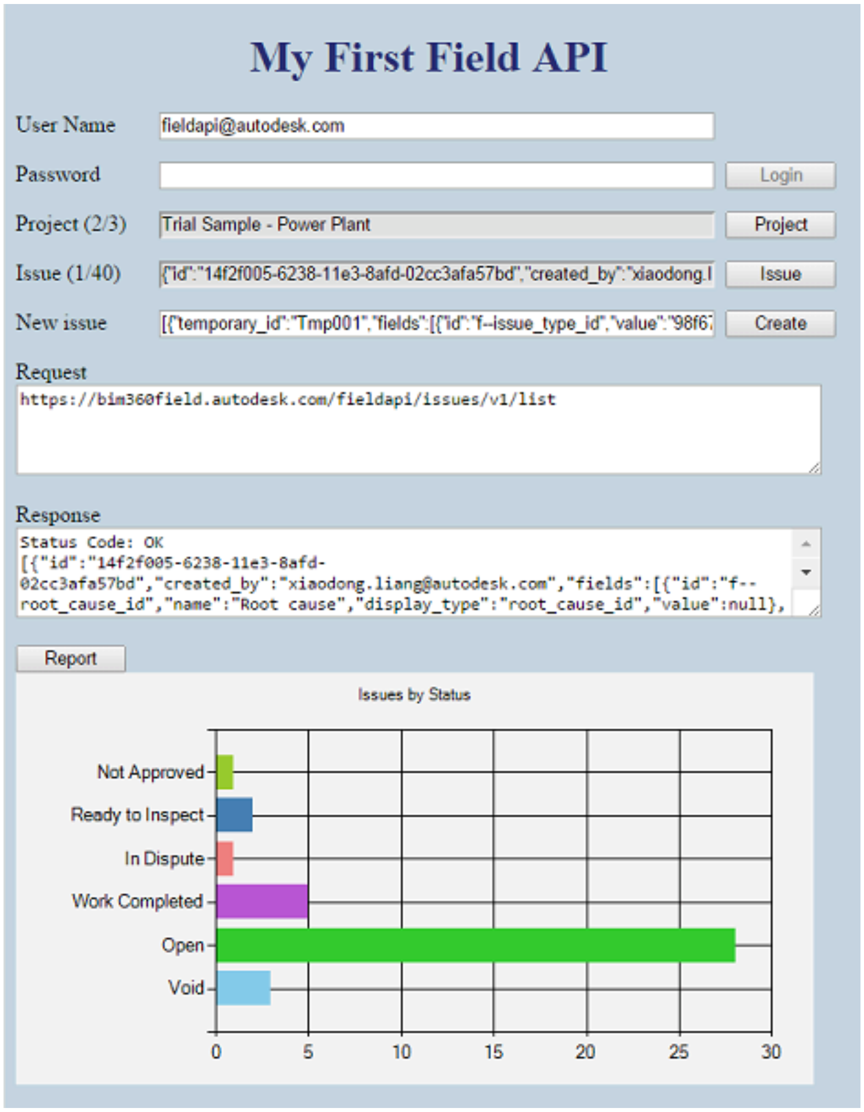
Using MSVS, you can use Data/Chart tool to add a chart to your web page. Simply drag and drop a tool to the web form. It will automatically place a chart <asp:Chart/> on your web page. You can adjust color and other details for the appearance of the chart in the property window. To change the type of chart, go to “Series” property; in the Series Collection Editor, change the ChartType, for example, to “Bar”.



Actual data goes to Chart >> Series >> Points. The below is an example of code that creates a chart when a “Report” button is clicked:

```
protected void ButtonReport_Click(object sender, EventArgs e)
 {
     List<Issue> issue_list = Session["issueList"] as List<Issue>;

     // Collect data from the issue list.
     // Count the number of issues for each status.
     Dictionary<string, int> data = CollectData(issue_list);

     // Clear the chart data
     Chart1.Series[0].Points.Clear();

     // Fill the chart data
     foreach (var item in data)
     {
         Chart1.Series[0].Points.AddXY(item.Key, item.Value);
     }

     // Add a title text
     Chart1.Titles[0].Text = "Issues by Status";

 }
```

An example of final web page will look like this:



My First Field API

| | |
|---|---|
| User Name | fieldapi@autodesk.com |
| Password | |
| Project (2/3) | Trial Sample - Power Plant |
| Issue (1/40) | {"id":"14f2f005-6238-11e3-8afd-02cc3afa57bd","created_by":"xiaodong.l |
| New issue | [{"temporary_id":"Tmp001","fields":[{"id":"f--issue_type_id","value":"98f6 |

Request

```
https://bim360field.autodesk.com/fieldapi/issues/v1/list
```

Response

```
Status Code: OK
[{"id":"14f2f005-6238-11e3-8afd-
02cc3afa57bd","created_by":"xiaodong.liang@autodesk.com","fields":[{"id":"f--
root_cause_id","name":"Root cause","display_type":"root_cause_id","value":null},
```

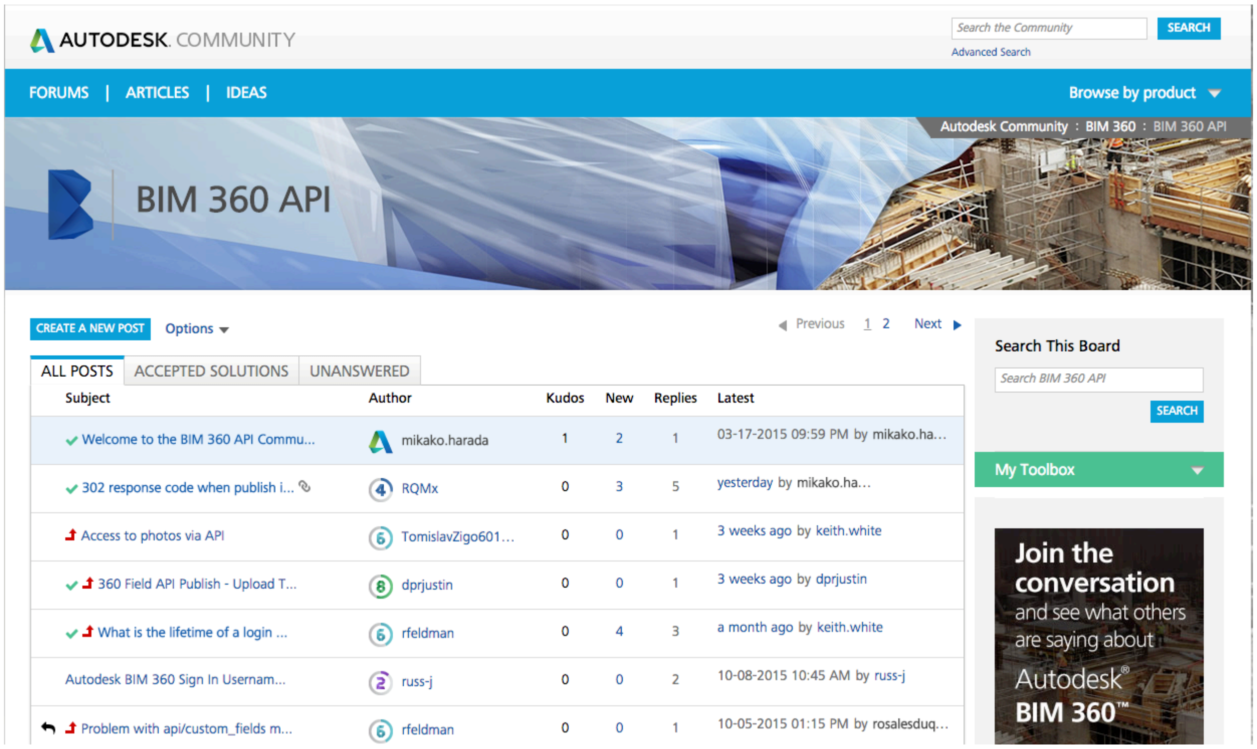
## Learn useful resources for the BIM 360 Field API

Some of the best resources for you to learn more about the Field API can be found online. Where material on where the API is heading and what are the new functionalities available can be found.

**BIM 360 API Forum**

**http://forums.autodesk.com/t5/bim-360-api/bd-p/115**

The dedicated BIM 360 API Forum for example, gives you the chance to ask questions on current workflows or blockers you may have, where API Autodesk and 3rd party API developers will have access and help the community with answers to the questions you may have.



| Subject | Author | Kudos | New | Replies | Latest |
|---|---|---|---|---|---|
| ✓ Welcome to the BIM 360 API Commu... | mikako.harada | 1 | 2 | 1 | 03-17-2015 09:59 PM by mikako.ha... |
| ✓ 302 response code when publish i... | RQMx | 0 | 3 | 5 | yesterday by mikako.ha... |
| Access to photos via API | TomislavZigo601... | 0 | 0 | 1 | 3 weeks ago by keith.white |
| ✓ 360 Field API Publish - Upload T... | dprjustin | 0 | 0 | 1 | 3 weeks ago by dprjustin |
| ✓ What is the lifetime of a login ... | rfeldman | 0 | 4 | 3 | a month ago by keith.white |
| Autodesk BIM 360 Sign In Usernam... | russ-j | 0 | 0 | 2 | 10-08-2015 10:45 AM by russ-j |
| Problem with api/custom_fields m... | rfeldman | 0 | 0 | 1 | 10-05-2015 01:15 PM by rosalesduq... |

**The 360 View Blog**

http://the360view.typepad.com/blog/

The vision for this blog is aligned with the vision of BIM 360 family within Autodesk to "deliver complete, accurate, connected, digital information across the project lifecycle for whoever needs it, whenever they need it, wherever they are."

This blogs aims to be a technical resource for customers, partners, and developers interested in leveraging the BIM 360 cloud based services and applications, and complements the information available at our Construction Solutions blog - beyond design, and developer blogs by Autodesk Developer Network - ADN devblogs, etc. In addition to a continued focus on innovation and best practices, this blog will look to highlight the data integrations and cloud based workflows based on BIM 360 platform. We will be posting about the possibilities of extending BIM 360 functionalities through success stories, new APIs, code samples, and other interesting snippets. If you are a BIM 360 user, or developer or a decision maker looking to learn more about the possibilities, this might be a good resource.

The 360 view

Democratizing BIM in the cloud

Home Subscribe Autodesk 360 blog

ABOUT THE AUTHORS

Manu Venugopal Mikako Harada

Sheryl Yang

OCTOBER 2015

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

CATEGORIES

ADN API augmented reality BIM BIM 360 Cloud Construction Dashboards Data Integration Design Computation DevDays Field Generative Design Glue Innovation Intelligence javascript Lean Production Planning REST Viewer Virtualization

10/07/2015

**Autodesk 'BIM for Owners' Event**

Autodesk recently hosted an event on 'BIM for Owners' in NYC at September, 17th. The objective of the one-day workshop was to provide an opportunity for project management professionals within owner's organizations to discuss and exchange what they understand about BIM, what are the benefits of using BIM for owners,... Read more →

ShareThis Facebook Tweet LinkedIn Email

Posted at 10:35 AM in BIM | Permalink | Comments (0)

09/03/2015

**Adding High Level Category to the Issue Type**

Q. Is there a way to add a new high level category to the issue type using Field API? A. Please try: POST /fieldapi/issues/v1/create_type This method takes a name of category and a name of issue type: ticket project_id category_name – string. The name of category (e.g., Punch List, Safety)... Read more →

ShareThis Facebook Tweet LinkedIn Email

Posted at 08:46 PM in API, Field, REST | Permalink | Comments (0)

Subscribe

AUTODESK® BIM 360™

Knowledge Network

BIM 360 Portal

BIM 360 API Forum

BIM 360 Glue Help

BIM 360 Field Help

Free Trial

Try BIM 360 Glue

Try BIM 360 Field

SEARCH

Search

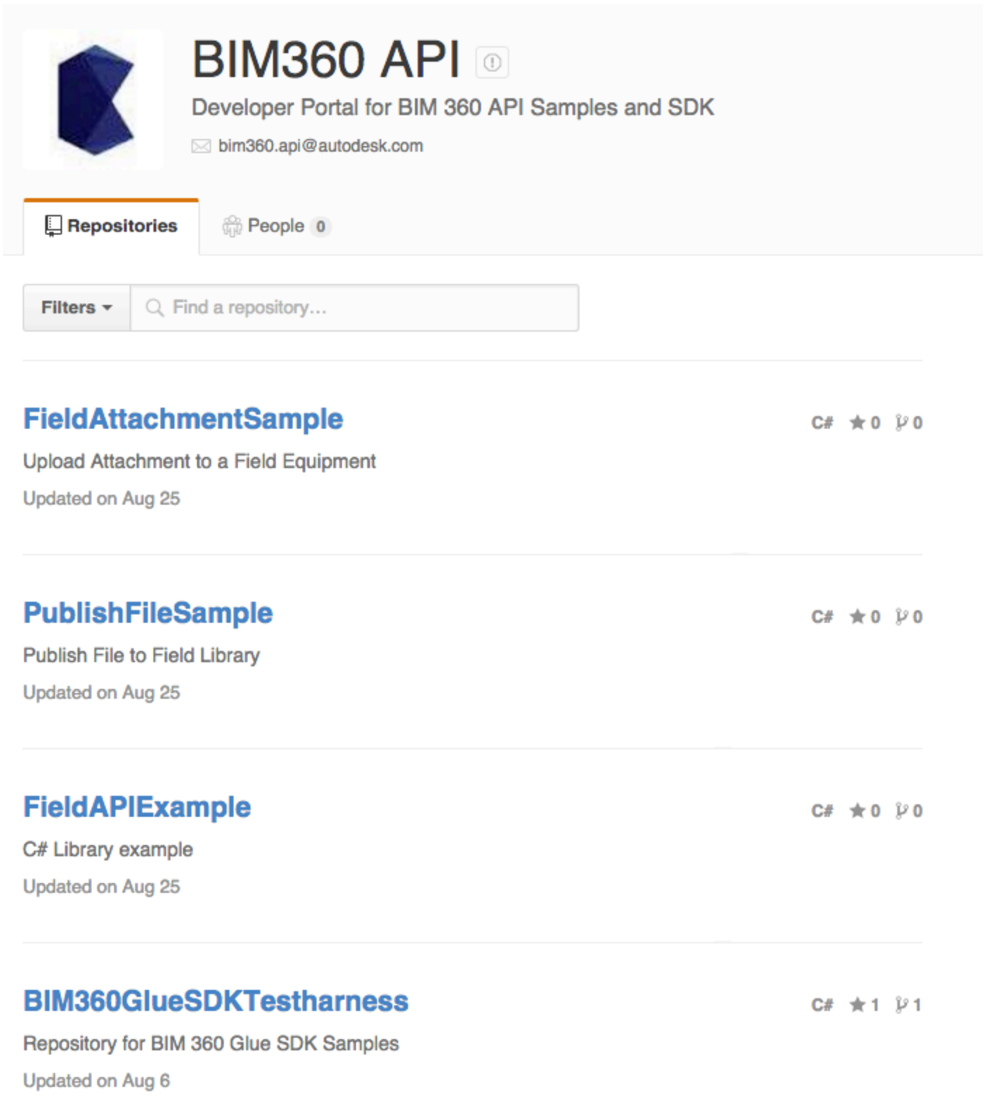
RECENT POSTS

Autodesk 'BIM for Owners' Event

**GitHub Repository – BIM 360 API**

https://github.com/BIM360API

Not too long ago, a repo dedicated to the BIM 360 API was released. Currently contains some of the samples previously shown in this class, but also you can find other helpful ones like the Un-Official TestHarness project where you can explore many of the Field API calls at a more deeper level. Fork the Repos, give it a try and expand your Field Applications to a higher level.


BIM360 API

Developer Portal for BIM 360 API Samples and SDK

bim360.api@autodesk.com

**FieldAttachmentSample** C# ★ 0 ⑂ 0

Upload Attachment to a Field Equipment

Updated on Aug 25

**PublishFileSample** C# ★ 0 ⑂ 0

Publish File to Field Library

Updated on Aug 25

**FieldAPIExample** C# ★ 0 ⑂ 0

C# Library example

Updated on Aug 25

**BIM360GlueSDKTestharness** C# ★ 1 ⑂ 1

Repository for BIM 360 Glue SDK Samples

Updated on Aug 6