

CS125381

## Safety First—with BIM 360 Field

Lauren L Collier  
SSOE Group

### Learning Objectives

- Learn how to create your own custom checklist in BIM 360 Field
- Learn how to use and replicate a practical safety Checklist example
- Understand what data is available to connect and harvest from BIM 360 Field
- Learn how to visualize and make project safety decisions based on live project data

### Description

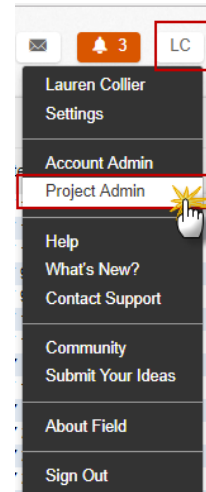
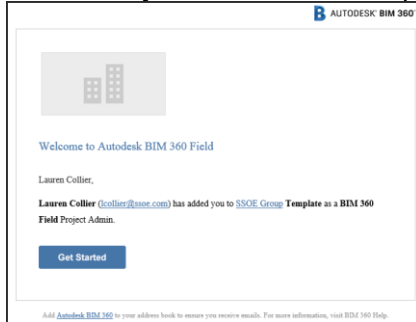
This class will “safely” introduce how to integrate BIM 360 Field software into a construction safety program. If your company is looking to start using BIM 360 Field, safety is an excellent and safe approach to beginning. We’ll explore the creation of custom checklists, and how to create templates for consistency across projects. The presentation will guide you through the application, and we’ll provide some tips and tricks to being successful in data collection. A Safety First culture is to be both preventative and proactive on our job sites—not only is it significant to be rigorous in monitoring, but what do you do with all this data collected? The second part of the presentation will look at the open BIM 360 API. We’ll share live project dashboards to replace your stagnant weekly or monthly summaries—but, more importantly, to make safer decisions for your projects. The activity trends are now more obvious and visible to the construction teams, letting both praise and corrective actions occur.

### Speaker(s)

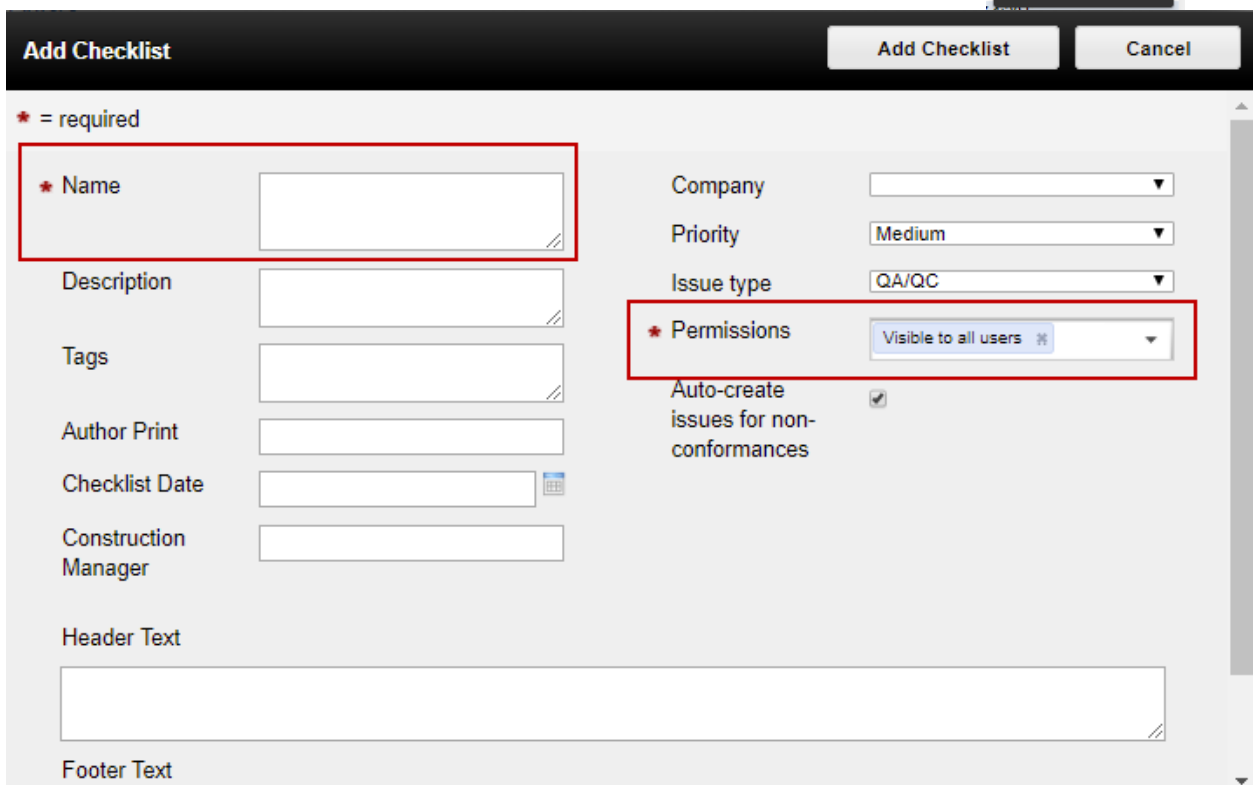
Lauren has over 12 years of industry experience ranging from healthcare and industrial architectural design, BIM implementation, Innovation Projects and executing Corporate Initiatives. She holds a Master of Architecture degree from Savannah College of Art and Design and a Bachelor of Art and Architecture degree from Miami University. Her creative passion lies in Lean, constant improvements and innovative model/data use solutions for design and construction operations. She leads a group of VDC Technical leaders and Model Managers whose focus is implementing these new innovative technologies and best practices in design and construction. In addition to her role at SSOE, Lauren is an active amateur photographer as well as a wife and mother of two imaginative and energetic boys.

## Creating a Checklist

1. Log into Web BIM 360 Field Classic – <https://bim360field.autodesk.com/home>
2. Reminder you must have accepted an invite to Project.



3. Go to User Initials Far Right of Web Page
4. Go to Project Admin
5. Checklist Icon – Pick Type: QA/QC; Safety, Commissioning.



7. Users can Import from System, User defined, or create from Scratch.



- 8.

Import Checklists

ImportClose

From Account

From File

<input type="checkbox"/>	Name	# items	Project	Status
<input type="checkbox"/>	Hazards			
<input type="checkbox"/>	OHSAS - IDENTIFIED HAZARDS	31	Vela Samples	Active
<input type="checkbox"/>	OSHA 3216-6N-06 - CONSTRUCTION HAZARDS	19	Vela Samples	Active
<input type="checkbox"/>	OSHA 3216-6N-06 - CONSTRUCTION HAZARDS	19	Template	Active

9.  
10. Import Option

Import Checklists

Close

From Account

From File

Imported files should be formatted as follows, with one checklist item per row:

	A	B	C	D	E	F	G	H	I	
1	Checklist Name	Display Number	Question Text	Response Type	Default Answer	Required	Responsible Cc	Details	Spec Ref	
2	Sample Checklist		Sample Group   Group Header							
3	Sample Checkli	1	Ensure bolts an Plus, Minus, N/	N/A		0		Sample instruct	Division 09	Finishes
4	Sample Checkli	2	Verify type, size Yes, No, N/A	N/A		0		Sample instruct	Division 09	Finishes
5	Sample Checkli	3	Verify all heads True, False, N/	N/A		0		Sample instruct	Division 09	Finishes

[Download a sample](#)

Great Starting Point for Large Audits

**Note** - Format dates as follows: 2017-11-07

Choose an Excel file to import below.

Then, click **Start Import** to begin importing.

Choose File

No file chosen

Start Import

11. Add Custom Properties

QA/QC

Safety

Commissioning

Standard Properties

Custom Properties

Templates

+ Add

Edit

Delete

Name	Type	Default Value	Source	Required
All Templates				
Safety-Safety 1926 Subpart T - Demolition				
Safety-SSOE_Job_Site_Safety_Checklist				
<input type="checkbox"/> Label	Text		Custom	<input checked="" type="checkbox"/>
<input type="checkbox"/> Author Print	Text		Custom	<input checked="" type="checkbox"/>
<input type="checkbox"/> Author Signature	Signature		Custom	<input checked="" type="checkbox"/>
<input type="checkbox"/> Checklist Date	Date		Custom	<input checked="" type="checkbox"/>
<input type="checkbox"/> Construction Manager	Text		Custom	<input checked="" type="checkbox"/>

12.

### 13. Create from Scratch

Add Checklist
Add Checklist
Cancel

\* = required

\* Name
Description
Tags
Author Print
Checklist Date
Construction Manager
Header Text
Footer Text

Company
Priority
Issue type
\* Permissions
Auto-create issues for non-conformances

### 14. Add Section

Add Checklist Section
Add Checklist Section
Cancel

\* = required

\* Section Name
Section Description

### 15. Add Items

Add Checklist Item
Add Checklist Item
Cancel

Item
More Info

\* = required

Number
\* Item Text

Response
Type

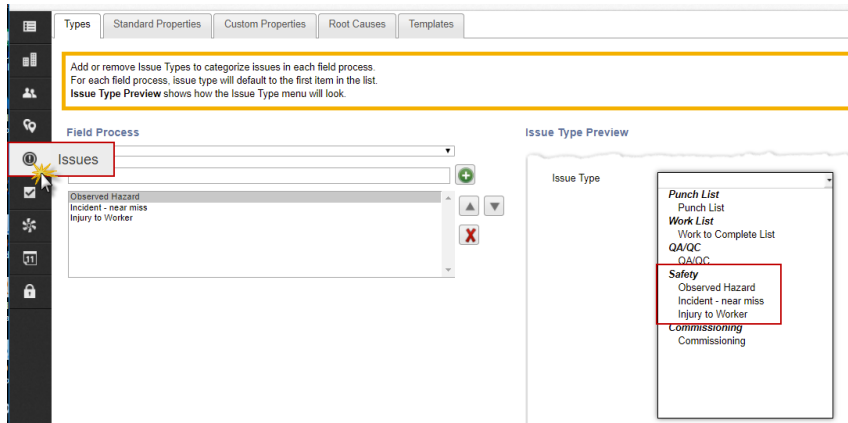
Issue creation
Default values for issues created from this checklist item:
Description
Root cause
Company
Spec reference

16. Recommendation – You can't edit Response Types once created. Users have to re-do.

## Additional Basic Set-up

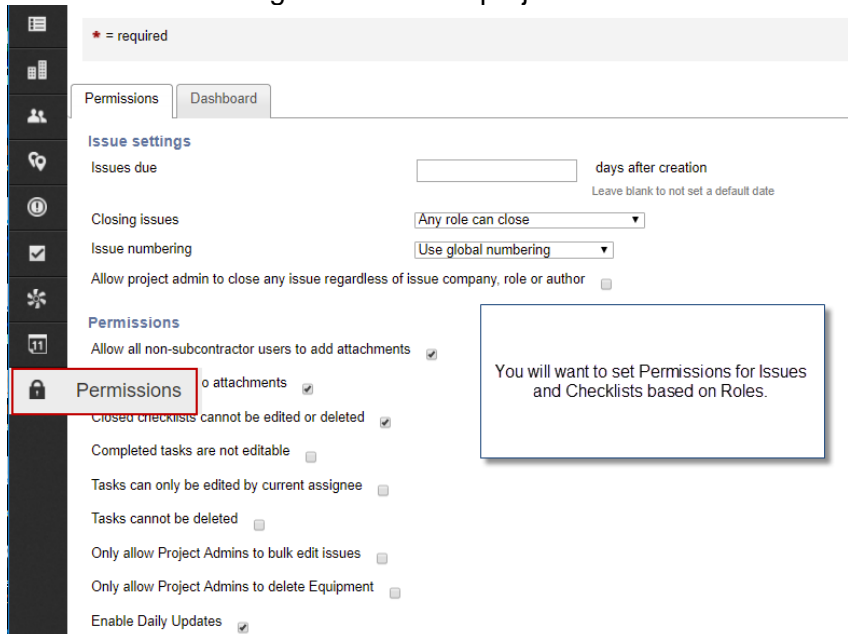
### Issues

- Set Types



### Permissions

- Create default settings and roles for project



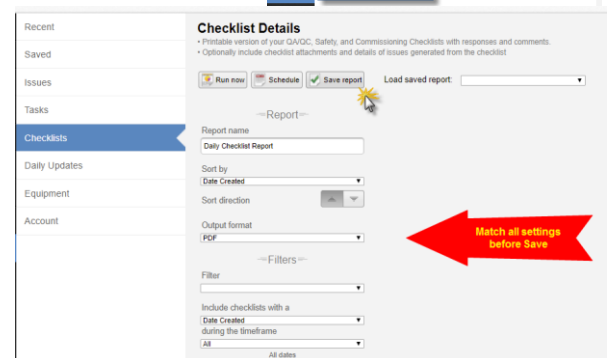
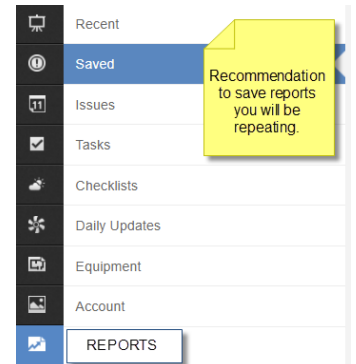
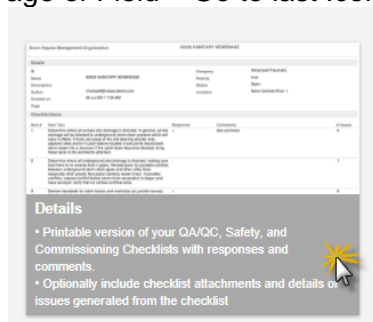
Functions	Contractor	Architect	Engineer	Owner	Inspector
Issues	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>
QA/QC Checklists	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>
Safety Checklists	<a href="#">View+Add+Edit</a>	<a href="#">View+Add</a>	<a href="#">View+Add</a>	<a href="#">View+Add</a>	<a href="#">View+Add</a>
Daily Updates	<a href="#">My Own</a>	<a href="#">My Own</a>	<a href="#">My Own</a>	<a href="#">My Own</a>	<a href="#">My Own</a>
Equipment (and Cx Checklists)	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>	<a href="#">View+Add+Edit</a>
Run Reports	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enforce Role Filters in Reports	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Library (View + Markup)	<a href="#">View+Markup</a>	<a href="#">View+Markup</a>	<a href="#">View+Markup</a>	<a href="#">View+Markup</a>	<a href="#">View+Markup</a>
Assign Tasks	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

## Add Companies & Members

- Add Companies who will be on site
- You can have a company added to project without members
- This is useful for dailies, issue and task assignments.
- Add Members who will be viewer, using, or interacting with Field
- Apply Roles to users to set permissions

## Create Saved Reports

- In Main Page of Field – Go to last Icon Reports.



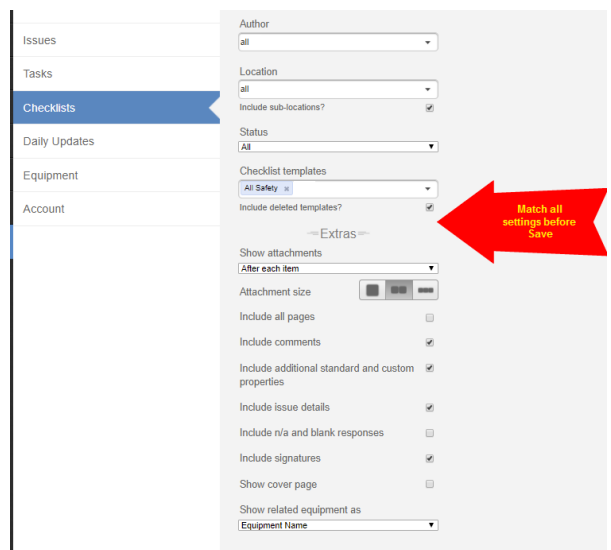
## Tips and Tricks for Field

### Recommendations

- Plan out Sections of your Audits
- Be consistent in your Response Types
- Plan which checklist templates to be added to project
- Have Checklist exports for easy future editing.
- Create Saved Reports for Issues and Checklists.

### Warnings

- Create a Multi-List Answer that needs updates a lot. Too much administration
- Manage Custom Properties on Unique Checklists
- Too Large of Audits are difficult to Scroll on iPad App



## API and Data Harvesting

API Documentation Link: [https://bim360field.autodesk.com/apidoc/index.html#mobile\\_api\\_method\\_1](https://bim360field.autodesk.com/apidoc/index.html#mobile_api_method_1)  
 SSOE wrote a Custom BIM 360 Harvester Application with the Rest API.

### Get

```

//Harvester
public class API
{
    private string _url;
    private Token _token;
    private Project _defaultProject;
    private string _login = "/api/login";
    private string _logout = "/api/logout";
    private string _projects = "/api/projects";
    private string _project = "/api/project";
    private string _templates = "/api/templates";
    private string _contacts = "/api/contacts";
    private string _get_tasks = "/api/get_tasks";
    private string _checklist = "/fieldapi/checklists/v1"; // I'm RESTful (but, not all operations are supported!)
    private string _create_checklist = "/api/checklists";
    private string _get_checklists = "/api/get_checklists";
    private string _get_equipment = "/api/get_equipment";
    private string _issues = "/fieldapi/issues/v2"; // I'm RESTful
    private string _get_locations = "/fieldapi/admin/v1/locations";
    private string _get_companes = "/fieldapi/admin/v1/companies";
    private string _get_companies = "/fieldapi/companies/v1";
    //private string _post_companies = "/fieldapi/companies/v1";
    private string _get_issue_types = "/fieldapi/issues/v1/types";
    private string _create_issue_type = "/fieldapi/issues/v1/create_type";
    private string _destroy_issue_type = "/fieldapi/issues/v1/destroy_type";
    private string _create_issue = "/fieldapi/issues/v1/create";
    private string _get_issues = "/api/get_issues";
    private string _issue_filters = "/api/issue_filters";
    private string _get_users = "/fieldapi/admin/v1/users";
    private string _publish = "/api/library/publish";
    private string _delete_document = "/api/library/delete";
    private string _all_folders = "/api/library/all_folders";
    private string _all_files = "/api/library/all_files";
    private string _custom_fields = "/api/custom_fields";
    private string _create_custom_field = "/fieldapi/admin/v1/custom_field_create";
    private string _vela_fields = "/api/vela_fields";
    private string _binary_data = "/api/binary_data";
    private string _create_project = "/fieldapi/admin/v1/project_create";
    private string _attachment = "/api/attachments";
  }

```

- Project

```

//Harvester
public void GetProjects() //Getting Projects
{
    try
    {
        Console.WriteLine("Reading Project List");

        Projects = api.getProjects();

        if (Projects != null)
        {
            foreach (Project prj in Projects)
            {
                Console.WriteLine("    " + prj.name);
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Failed to get Project List : " + ex.Message);
    }
}

```

- Companies

```
public void WriteCompanies(SqlConnection cn)
{
    try
    {
        DeleteTbl(cn, "Companies");
        using (var cmd = new SqlCommand("select * FROM Companies", cn))
        {
            using (var sqlda = new SqlDataAdapter(cmd))
            {
                using (var sqlds = new DataSet())
                {
                    sqlda.Fill(sqlds);
                    if (sqlds.Tables.Count == 0) return;
                    DataTable dt = sqlds.Tables[0];

                    List<Company> comp = api.getCompanies();

                    if (comp != null)
                    {
                        foreach (Company cmp in comp)
                        {
                            DataRow dr = dt.NewRow();
                            dr["Company_ID"] = cmp.id;
                            dr["Company_name"] = cmp.name;
                            dr["Company_Description"] = cmp.description;
                            dr["Company_city"] = cmp.address.city;
                            dr["Company_state"] = cmp.address.state;
                            dr["Company_zip"] = cmp.address.postal_code;
                            dr["Country"] = cmp.address.country;
                            dr["Company_Phno"] = cmp.telephone;
                            dt.Rows.Add(dr);
                        }
                        //write data
                        using (var cb = new SqlCommandBuilder(sqlda))
                        {
                            sqlda.Update(sqlds);
                        }
                    }
                }
            }
        }
    }
}
```

- Checklists

```
public void GetCheckLists()
{
    Console.WriteLine("Reading Checklists");

    Checklist tmpchk;

    Checklists = new List<Checklist>();

    try
    {
        using (var SW = new StreamWriter(@"FileDownloads.csv"))
        {
            foreach (Project prj in Projects)
            {
                Console.WriteLine("    " + prj.name);

                List<Checklist> tmp = api.getChecklists(null, prj.project_id, 0, 200); //get the checklist list
                if (tmp != null)
                {
                    foreach (Checklist chk in tmp) //loop through each checklist in the list of checklists tmp
                    {
                        tmpchk = api.getChecklist(chk.id, prj.project_id);

                        if (tmpchk != null)
                        {
                            Console.WriteLine("        Reading : " + " , " + tmpchk.identifier + " , " + tmpchk.name + " , " + tmpchk.updated_at + " , " + tmpchk.getchecklist_date());
                            Checklists.Add(tmpchk); //add each checklist in list of checklists
                        }
                        WriteChecklistData(tmpchk, SW);
                    }
                }
            }
        }
    }
}
```

```

public void WriteChecklists(SqlConnection cn)
{
    try
    {
        using (var cmd = new SqlCommand("select * FROM Checklists WHERE 0=1", cn))
        {
            using (var sqlda = new SqlDataAdapter(cmd))
            {
                using (var sqlds = new DataSet())
                {
                    sqlda.Fill(sqlds);
                    if (sqlds.Tables.Count == 0) return;
                    DataTable dt = sqlds.Tables[0];

                    //Fill dt with Data

                    foreach (Checklist chk in Checklists)
                    {
                        DataRow dr = dt.NewRow();

                        dr["Checklist_ID"] = string.IsNullOrEmpty(chk.id) ? Guid.Empty.ToString() : chk.id; // don not need to for question_ID
                        dr["Project_ID"] = string.IsNullOrEmpty(chk.project_id) ? Guid.Empty.ToString() : chk.project_id;
                        dr["Checklist_name"] = chk.name;
                        dr["Checklist_Type"] = chk.checklist_type;
                        dr["Priority"] = chk.priority;
                        dr["Status"] = chk.status;
                        dr["Company_ID"] = string.IsNullOrEmpty(chk.company_id) ? Guid.Empty.ToString() : chk.company_id;
                        dr["DateCreated"] = chk.created_at;
                        DateTime cId = chk.getchecklist_date();

                        if (cId.Ticks > 0) dr["ChecklistDate"] = cId; //If the time is greater than 0, then write the actual value to data base.

                        dt.Rows.Add(dr);
                    }
                }
            }
        }
    }
}

```

- Questions
- Issues

```

public void GetIssuelist()
{
    Console.WriteLine("Reading Issues");

    Issue tmpIss;

    Issues = new List<Issue>();

    try
    {
        using (var SW = new StreamWriter(@"issuedownloads.csv"))
        {
            foreach (Project prj in Projects)
            {
                Console.WriteLine("    " + prj.name);

                List<Issue> tmp = api.getIssuelist(null, null, prj.project_id);

                if (tmp != null)
                {
                    foreach (Issue Iss in tmp)
                    {
                        Console.WriteLine("        Reading : " + " , " + Iss.identifier + " , " + Iss.description);
                        tmpIss = api.getIssue(null, prj.project_id);
                        if (tmpIss != null)
                            Issues.Add(tmpIss);
                    }
                }
            }
        }
    }
}

```

```
public void WriteIssues(SqlConnection cn)
{
    try
    {
        using (var cmd = new SqlCommand("select * FROM Issues WHERE 0=1", cn))
        {
            using (var sqlda = new SqlDataAdapter(cmd))
            {
                using (var sqlds = new DataSet())
                {
                    sqlda.Fill(sqlds);
                    if (sqlds.Tables.Count == 0) return;
                    DataTable dt = sqlds.Tables[0];

                    foreach (Project prj in Projects)
                    {
                        List<Issue> tmp = api.getIssuelist(null, null, prj.project_id);
                        if (tmp != null)
                        {
                            foreach (Issue Iss in tmp)
                            {
                                DataRow dr = dt.NewRow();
                                dr["Issue_ID"] = Iss.issue_id; // don not need to for question_ID
                                dr["Issue_Description"] = Iss.description;
                                dr["Checklist_ID"] = string.IsNullOrEmpty(Iss.source_id) ? Guid.Empty.ToString() : Iss.source_id;
                                dr["Company_ID"] = string.IsNullOrEmpty(Iss.company_id) ? Guid.Empty.ToString() : Iss.company_id;
                                dr["Project_ID"] = prj.project_id;
                                if (Iss.created_at.Ticks > 0) dr["dateCreated"] = Iss.created_at;
                                if (Iss.updated_at.Ticks > 0) dr["DateUpdated"] = Iss.updated_at;
                                if (Iss.due_date.Ticks > 0) dr["Issue_DueDate"] = Iss.DueDate;
                                dr["Issue_Status"] = Iss.status;
                                dt.Rows.Add(dr);
                            }
                        }
                    }
                }
            }
        }
    }
}
```

## Post

- We chose to Push our Company data.

### POST /api/companies

**Description:** Returns a list of companies in the specified project.

DEPRECATED .. backwards compatible support only! See GET /fieldapi/companies/v1

**Access:** FREE

**Return:** [JSON] - Returns an array of companies defined in the project specified.

**Parameters:**

**max\_date** : string - If specified, only retrieve companies that have been created or updated after this date.

**ticket** : string - The ticket obtained from the login call.

**project\_id** : string - The ID of the project to perform this action against.

- Our master vendors live in Deltek we update monthly or as needed to BIM 360 Enterprise Location.