

Contents

1 PL-400 Microsoft Power Platform Developer	5
1.1 What are we doing?	5
1.2 How should I use these files relative to the released MOC files?	5
1.3 What about changes to the student handbook?	5
1.4 How do I contribute?	5
1.5 Notes	5
1.5.1 Classroom Materials	5
1.6 It is strongly recommended that MCTs and Partners access these materials and in turn, provide them separately to students. Pointing students directly to GitHub to access Lab steps as part of an ongoing class will require them to access yet another UI as part of the course, contributing to a confusing experience for the student. An explanation to the student regarding why they are receiving separate Lab instructions can highlight the nature of an always-changing cloud-based interface and platform. Microsoft Learning support for accessing files on GitHub and support for navigation of the GitHub site is limited to MCTs teaching this course only.	5
1.7 title: Online Hosted Instructions permalink: index.html layout: home	5
2 Content Directory	5
2.1 Labs	6
2.2 lab: title: 'Lab 00: Validate lab environment'	6
2.3 Practice Lab – Validate lab environment	6
2.4 Scenario	6
2.5 Exercise 1 – Acquire your Power Platform trial tenant	6
2.6 Exercise 2 - Create your environment	6
2.6.1 Task 1 – Create environment	6
2.7 Exercise 3 - Azure DevOps account setup	7
2.8 lab: title: 'Lab 01: Data Modeling'	7
2.9 Lab 01 – Data Modeling	7
3 Scenario	7
4 High-level lab steps	8
4.1 Things to consider before you begin	8
5 Exercise #1: Create Environments and Solution	8
5.1 Task #1: Create Environments	8
5.2 Task #2: Create Solution and Publisher	9
5.3 Task #3: Add Existing Table	10
6 Exercise #2: Create Tables and Columns	11
6.1 Task #1: Create Permit Table and Columns	11
6.2 Task #2: Create Permit Type Table and Columns	13
6.3 Task #3: Create Build Site Table and Columns	13
6.4 Task #4: Create Inspection Table and Columns	16
6.5 Task #5: Edit Status Reason Options	18
7 Exercise #3: Create Relationships	21
7.1 Task #1: Create Relationships	21
8	24
8.1 lab: title: 'Lab 02: Model driven app'	24
8.2 Lab 02 – Model driven app	25
9 Scenario	25
10 High-level lab steps	25
10.1 Things to consider before you begin	25
11 Exercise #1: Customize Views and Forms	26
11.1 Task #1: Edit Permit Form and View	26
11.2 Task #2: Edit Build Site Form and View	31

11.3 Task #3: Edit Inspection Form and View	32
11.4 Task #4: Edit Permit Type Form	36
12 Exercise #2: Create Model-Driven Application	37
12.1 Task #1: Create Application	37
12.2 Task #2: Test Application	41
12.3 lab: title: 'Lab 03: Canvas app'	45
12.4 Lab 03 – Canvas app	45
13 Scenario	45
14 High-level lab steps	45
14.1 Things to consider before you begin	46
15 Exercise #1: Create Canvas App	46
15.1 Task #1: Create Canvas App	46
15.2 Task #2: Add Data Source	50
15.3 Task #3: Add Inspection Gallery	51
15.4 Task #4: Add Inspection Details Screen	54
15.5 Task #5: Submit the Inspection Result	62
15.6 Task #6: Test Application	63
16 Exercise #2: Export/Import Solution	66
16.1 Task #1: Export solution.	66
16.2 Task #2: Import solution.	70
16.3 lab: title: 'Lab 04: Client Scripting'	71
16.4 Lab 04 – Client Scripting	71
17 Scenario	71
17.1 High-level lab steps	72
17.2 Things to consider before you begin	72
18 Exercise #1: Prepare and Load Resources	72
18.1 Task #1: Use Visual Studio Code to Create Resources	72
18.2 Task #2: Add Event Handlers	74
18.3 Task #3: Load Web Resources	75
18.4 Task #4: Test Event Handlers	81
19 Exercise #2: Show and Hide Tabs	83
19.1 Task #1: Create Function	83
19.2 Task #2: Get Inspection Type Record	85
19.3 Task #3: Load Updated Script	87
19.4 Task #4: Test Your Changes	88
20 Exercise #3: Toggle *required property on the Columns	91
20.1 Task #1: Create Function	91
20.2 Task #2: Load Updated Script	94
20.3 Task #3: Test Your Changes	95
21 Exercise #4: Command Button Function	98
21.1 Task #1: Download and Install Ribbon Workbench	98
21.2 Task #2: Create Action Process	102
21.3 Task #3: Create the Function	104
21.4 Task #4: Add Button to Ribbon	108
21.5 Task #5: Test Command Button	112
21.6 lab: title: 'Lab 05: Power Apps Component Framework'	113
21.7 Lab 05 – Power Apps Component Framework	113
22 Scenario	113
23 High-level lab steps	113
23.1 Things to consider before you begin	114

24 Exercise #1: Create the PCF Control	114
24.1 Task #1: Install Microsoft Power Apps CLI and Prerequisites	114
24.2 Task #2: Setup Components Project	115
24.3 Task #3: Build the Basic Timeline	120
24.4 Task #4: Tailor for Inspection Data	125
24.5 Task #5: Change Color for Items	129
25 Exercise #2: Publish to Microsoft Dataverse	131
25.1 Task #1: Setup and Publish	131
25.2 Task #2: Add Timeline Control to the Permit Form	132
25.3 Task #3: Debug	138
25.4 Task #4: Add the Timeline Control to Permit Management Solution	141
26 Exercise #3: Promote to production	142
26.1 Task #1: Export Solution	142
26.2 Task #2: Import Solution	144
26.3 Review the production application by adding a few records and testing your progress.	145
26.4 lab: title: 'Lab 06: Plug-ins'	145
26.5 Lab 06 – Plug-ins	146
27 Scenario	146
28 High-level lab steps	146
28.1 Things to consider before you begin	146
29 Exercise #1: Block New Permit Creation Plugin	146
29.1 Task #1: Create the plugin	146
29.2 Task #2: Deploy the plugin	151
30 Exercise #2: Create Custom Action Plugin	157
30.1 Task #1: Add a new plugin to the project	158
30.2 Task #2: Get Related Inspections and Cancel	159
30.3 Task #3: Set Output Parameter and Create Note Record	161
30.4 Task #4: Deploy Plugin	162
31 Exercise #3: Test Plugins	166
31.1 Task #1: Test Lock Plugin	166
31.2 Task #2: Test Restrict New Permit Creation Plugin	174
32 Exercise #4: Plugin Trace Log and Debugging	176
32.1 Task #1: Plugin Trace Log	176
32.2 Task #2: Debugging Plugins (Optional)	178
32.3 Follow these steps to debug your plugins https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/tutorial-debug-plug-in	178
32.4 lab: title: 'Lab 07: Azure Functions'	178
32.5 Lab 07 – Azure Functions	178
33 Scenario	178
34 High-level lab steps	178
34.1 Things to consider before you begin	178
35 Exercise #1: Configure an application user	179
35.1 Task #1: Register Azure AD Application	179
35.2 Task #2: Create Application User and Security Role	181
36 Exercise #2: Create Azure Function for Inspection Routing	188
36.1 Task #1: Create the Function	188
36.2 Task #2: Get Inspections and Users and Assign Inspections	198
37 Exercise #3: Publish and Test	201
37.1 Task #1: Publish to Azure	202

38 Exercise #4: Promote to production	210
38.1 Task #1: Export Solution	210
38.2 Task #2: Import Solution	212
38.3 lab: title: 'Lab 08: Publishing Events Externally'	213
38.4 Lab 08 – Publishing Events Externally	213
39 Scenario	213
40 High-level lab steps	213
40.1 Things to consider before you begin	213
41 Exercise #1: Create an Azure Function	214
41.1 Task #1: Create Azure Function App	214
41.2 Task #2: Create an Azure Function	215
42 Exercise #2: Configure Web Hook	220
42.1 Task #1: Configure publishing to a web hook	220
42.2 Task #2: Test the Web Hook	225
42.3 Task #3: Configure an Table image	227
42.4 Note: Technically, we have the data in the target object already. However, if there are plugins modifying the data, PostImage will contain the copy as recorded in Microsoft Dataverse while Target contains the data as submitted on Save. In addition to that, preimage contains data before the save operation took place.	230
42.5 lab: title: 'Lab 09: Custom Connector'	230
42.6 Lab 09 – Custom Connector	230
43 Scenario	230
44 High-level lab steps	231
44.1 Things to consider before you begin	231
45 Exercise #1: Create the Azure Function	231
45.1 Task #1: Create CPM Calculation Function	231
46 Exercise #2: Create the Custom Connector	236
46.1 Task #1: Create the Custom Connector	236
47 Exercise #3 Test Connector	244
47.1 Task #1: Test on Canvas App	244
47.2 Task #2: Test on Flow	254
47.3 lab: title: 'Lab 10: Application Lifecycle Management'	259
47.4 Lab 10 – Application Lifecycle Management	259
48 Scenario	259
49 High-level lab steps	260
49.1 Things to consider before you begin	260
50 Exercise #1: Initialize Azure DevOps	260
50.1 Task #1: Sign in for Azure DevOps	260
50.2 Task #2: Configure Power Apps ALM Tasks	261
51 Exercise #2: Build Export Pipeline	265
51.1 Task #1: Export the Solution	265
52 Exercise #3: Test the Pipeline	275
52.1 Task #1: Run the Pipeline	275
52.2 Task #2: Modify Solution	278
52.3 Task #3: Run Build Pipeline	280

1 PL-400 Microsoft Power Platform Developer

- **Download Latest Student Handbook and AllFiles Content**
- **Are you a MCT?** - Have a look at our [GitHub User Guide for MCTs](#)
- **Need to manually build the lab instructions?** - Instructions are available in the [MicrosoftLearning/Docker-Build](#) repository

1.1 What are we doing?

- To support this course, we will need to make frequent updates to the course content to keep it current with the Azure services used in the course. We are publishing the lab instructions and lab files on GitHub to allow for open contributions between the course authors and MCTs to keep the content current with changes in the Azure platform.
- We hope that this brings a sense of collaboration to the labs like we've never had before - when Azure changes and you find it first during a live delivery, go ahead and make an enhancement right in the lab source. Help your fellow MCTs.

1.2 How should I use these files relative to the released MOC files?

- The instructor handbook and PowerPoints are still going to be your primary source for teaching the course content.
- These files on GitHub are designed to be used in conjunction with the student handbook, but are in GitHub as a central repository so MCTs and course authors can have a shared source for the latest lab files.
- It will be recommended that for every delivery, trainers check GitHub for any changes that may have been made to support the latest Azure services, and get the latest files for their delivery.

1.3 What about changes to the student handbook?

- We will review the student handbook on a quarterly basis and update through the normal MOC release channels as needed.

1.4 How do I contribute?

- Any MCT can submit a pull request to the code or content in the GitHub repro, Microsoft and the course author will triage and include content and lab code changes as needed.
- You can submit bugs, changes, improvement and ideas. Find a new Azure feature before we have? Submit a new demo!

1.5 Notes

1.5.1 Classroom Materials

1.6 It is strongly recommended that MCTs and Partners access these materials and in turn, provide them separately to students. Pointing students directly to GitHub to access Lab steps as part of an ongoing class will require them to access yet another UI as part of the course, contributing to a confusing experience for the student. An explanation to the student regarding why they are receiving separate Lab instructions can highlight the nature of an always-changing cloud-based interface and platform. Microsoft Learning support for accessing files on GitHub and support for navigation of the GitHub site is limited to MCTs teaching this course only.

1.7 title: Online Hosted Instructions permalink: index.html layout: home

2 Content Directory

Hyperlinks to each of the lab exercises and demos are listed below.

2.1 Labs

```
{% assign labs = site.pages | where_exp:"page", "page.url contains '/Instructions/Labs'" %} | Lab | --- | {%- for activity in labs %} | {{ activity.lab.title }}{% if activity.lab.type %} - {{ activity.lab.type }}{% endif %}|(/home/ll/Azure_clone/Azure_new/PL-400_Microsoft-Power-Platform-Developer/{{ site.github.url }}{{ activity.url }}) | {% endfor %}
```

2.2 lab: title: 'Lab 00: Validate lab environment'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

2.3 Practice Lab – Validate lab environment

Attention to MCTs: Please make sure you are familiar with the [TrainerPrepGuide](#) for this course, especially the teaching tips and recommendations.

2.4 Scenario

In this Module 0 lab, you will acquire a Power Platform trial tenant, access the Power Platform admin center and setup your Azure DevOps account. In the admin center, we will create an individual environment for configuration during the course.

2.5 Exercise 1 – Acquire your Power Platform trial tenant

1. Copy your **Microsoft 365 credentials** from the Authorized Lab Hoster.
2. Navigate to <powerapps.microsoft.com> and click **Start free**.
3. Under **Work email**, enter the email address from your Microsoft 365 credentials.
4. You see a prompt that you have an existing account with Microsoft. Select **Sign in**.
5. Enter the password provided by the Authorized Lab Hoster.
6. Select **Yes** to stay signed in.

2.6 Exercise 2 - Create your environment

In this exercise, you will create your **Development** environment that you will do the majority of your lab work in.

2.6.1 Task 1 – Create environment

1. Access <https://admin.Powerplatform.microsoft.com> and log in with your Microsoft 365 credentials if prompted again.
2. Select **Environments** and click **+New**.
 - For **Name**, enter **[my initials] Dev** (Example: AJ Dev)
 - For **Type**, select **Trial**.
 - Change the toggle on **Create a database for this environment?** to **Yes**.
 - Leave all other selections as default and click **Next**.
 - On the next tab, leave all selections to default and click **Save**
3. Your **Dev** environment should now show in the list of Environments.

4. Your environment may take a few minutes to provision. Refresh the page if needed. When your environment is prepared, select your **Dev** environment by clicking on the ellipses next to its name to expand the drop down menu and select **Settings**.
5. Explore the different areas in **Settings** that you are interested in but do not make any changes yet.

2.7 Exercise 3 - Azure DevOps account setup

In this exercise, you will create your Azure DevOps account that you will be using in Lab10

1. Get a new Azure Pass (valid for 30-days) from the instructor or other source.
 2. Use a private browser session, go to Microsoftazurepass.com to redeem your Azure Pass using the Dynamics 365 (or M365) credentials provided to you). [Redeem a Microsoft Azure Pass](#) Follow the instructions for redemption.
 3. Using the same browser session, go to portal.azure.com, then search for “Azure DevOps”. In the resulting page, click Azure DevOps Organizations.
 4. Next, click on the link called “My Azure DevOps Organizations” (or navigate to <https://aex.dev.azure.com/>).
 5. In the drop down box on the left, choose Default, instead of “Microsoft Account”
 6. Create a new organization (find blue box in upper right-hand corner of the screen) using the Default directory. Provide a unique Azure DevOps Organization name like FL- PermitManagement (replace FL with your first and last initials), select your region and click Continue.
 7. Choose the newly created organization, then choose Organization settings on the left-hand side of the screen
 8. Navigate to Organization settings -> Billing -> Setup billing -> Select an Azure subscription, then select the Azure Pass subscription, then choose “MS Hosted CI/CD” and set the field “Paid parallel jobs” to 1. Then click SAVE in the blue box at the bottom.
 9. Wait at least 3 hours before using the CI/CD capabilities so that new settings are reflected in the back end. Otherwise you will still see the message “This agent is not running because you have reached the maximum number of requests...”.
 10. As an optional step, you can validate this by creating a new pre-defined project using the newly created org with billing enabled, using <https://azureddevopsdemogenerator.azurewebsites.net/>. Wait for some time before trying, then run a test build.
-

2.8 lab: title: 'Lab 01: Data Modeling'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

2.9 Lab 01 – Data Modeling

3 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and perform automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

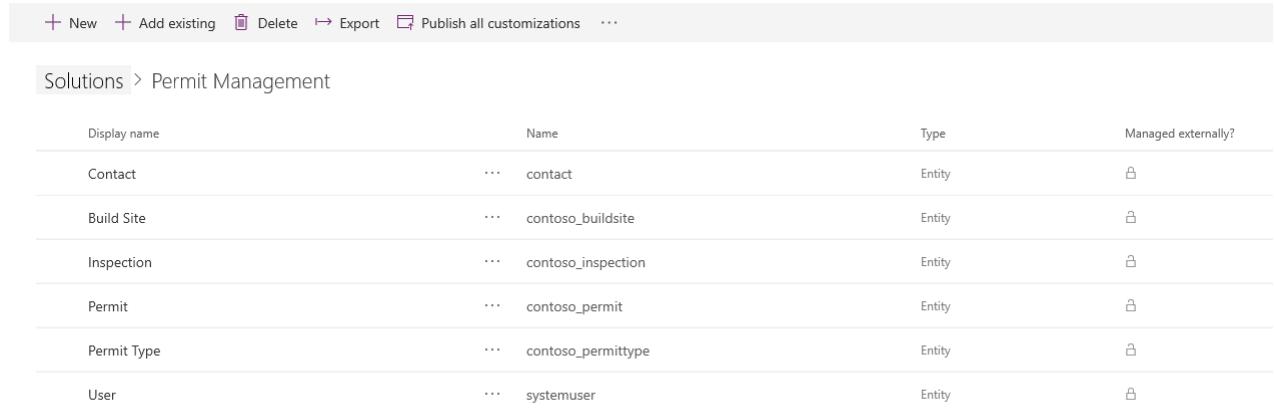
In this lab, you will set up a second environment to mimic a production environment for learning purposes, create solutions to track your changes. You will also create a data model to support the following requirements:

- R1 – Track the status of permits issued for new buildings and existing building modifications
- R2 – Permits are associated with a Build Site, which represents the building or land being modified

- R3 – Permit type indicates the type of permit and inspections, other data that might be required on a permit
- R4 – Inspections completed on the permit work are to be tracked for the entire process i.e., from request of inspection to the pass or fail of the inspection
- R5 – Permits, for our lab purposes, are requested by a person and we need to track who requested each permit

4 High-level lab steps

To prepare your learning environments you will create a solution and publisher and add both new and existing components that are necessary to meet the application requirements. Refer to the data model document for the metadata description (Tables, Column types and relationships). Your solution will contain several Tables upon completion of all the customizations.



The screenshot shows the Data Model view in Power Apps. At the top, there is a toolbar with buttons for New, Add existing, Delete, Export, Publish all customizations, and more. Below the toolbar, the breadcrumb navigation shows 'Solutions > Permit Management'. The main area displays a table of entities:

Display name	Name	Type	Managed externally?
Contact	... contact	Entity	□
Build Site	... contoso_buildsite	Entity	□
Inspection	... contoso_inspection	Entity	□
Permit	... contoso_permit	Entity	□
Permit Type	... contoso_permittype	Entity	□
User	... systemuser	Entity	□

4.1 Things to consider before you begin

- What are considered as best practices for managing changes in between environments (“Dev” to “Test” to “Prod”)? Are there additional considerations for team solution development?
- What Tables a user might need in the scenario that we are building?
- What relationship behaviors would we consider enabling users to complete their tasks?
- Remember to work in your **DEVELOPMENT** environment with the customizations. Once the customizations are completed, published and tested in “Dev”, and if everything works fine, the same will be deployed to “Prod”.

5 Exercise #1: Create Environments and Solution

Objective: In this exercise, you will create a community plan environment to mimic Production environment that we will refer to as “Prod”.

5.1 Task #1: Create Environments

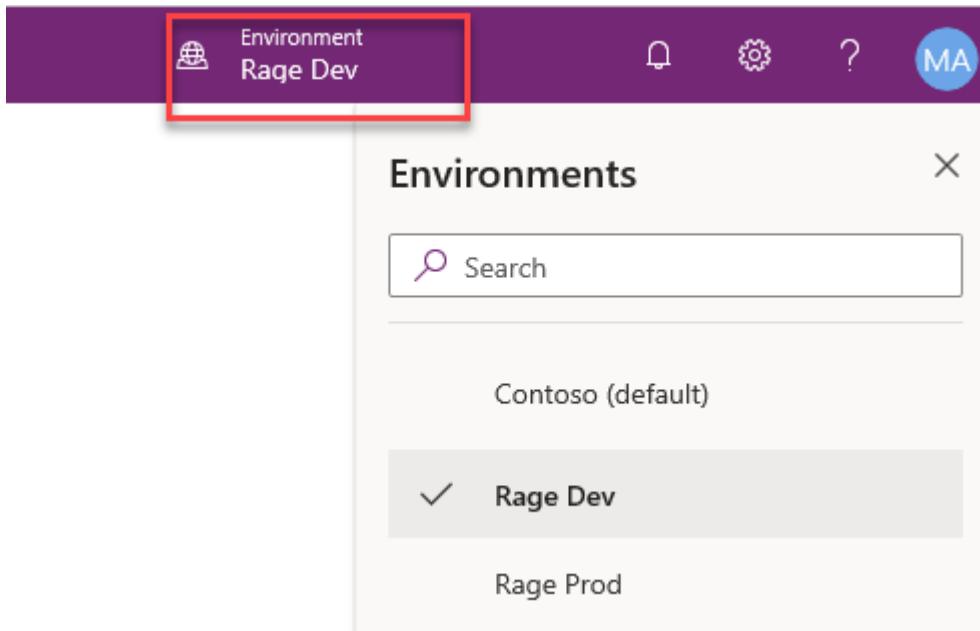
1. Create the community plan environment
 - Navigate to [Power Apps Community Plan page](#)
 - Click on *Create an individual environment*
 - Enter your credentials when prompt to sign in
 - Select your country from the dropdown menu and click *Accept*
 - Navigate to [Power Platform Admin Center](#) to see a new environment had been created by the system. We will refer to it as “Prod” environment for the rest of this course.

You should now have the dev environment and the “Prod” environment listed under environments.

5.2 Task #2: Create Solution and Publisher

1. Create Solution

- Sign in to [Power Apps maker portal](#)
- Select your Dev environment.



- Select **Solutions** from the left menu and click **+ New solution**.
- Enter **Permit Management** for **Display Name**.

2. Create Publisher

- Click on the **Publisher** dropdown and select **+ Publisher**.

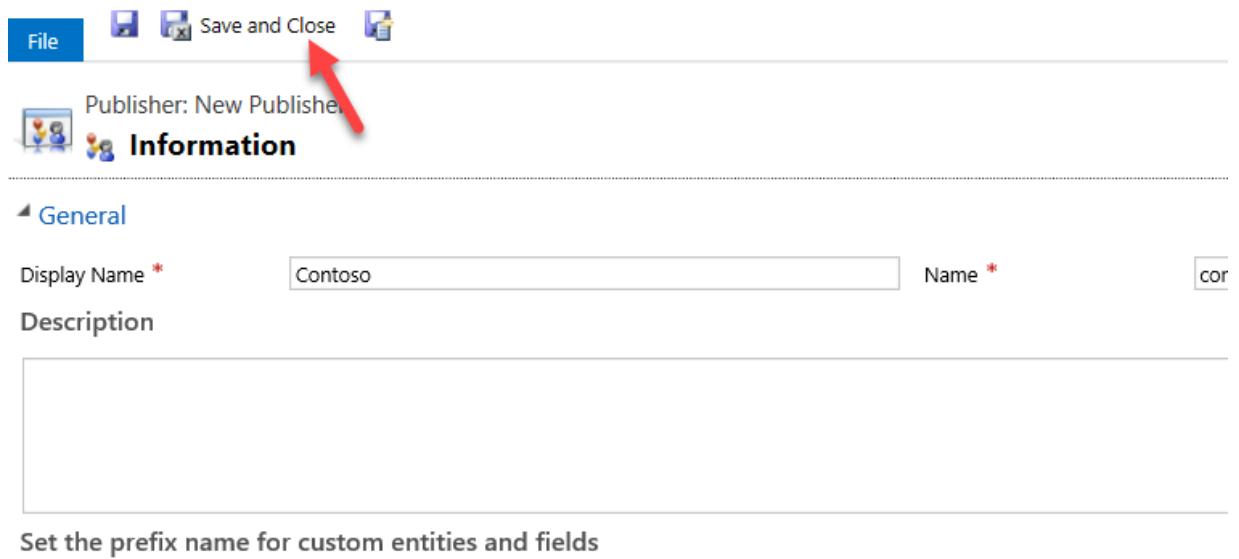
Name *

PermitManagement

Publisher *



- Enter **Contoso** for **Display Name** and **contoso** for **Prefix**.
- Click **Save and Close**.



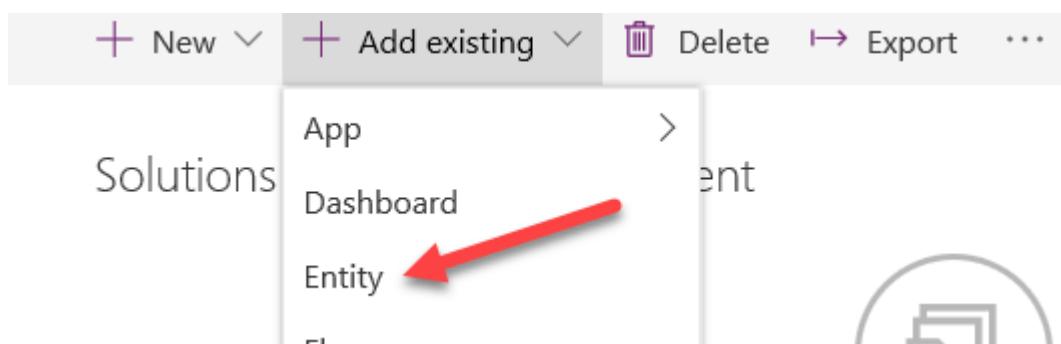
3. Complete the solution creation

- Now, click on the **Publisher** dropdown and select the **Contoso** publisher you just created.
- Enter **1.0.0.0** for **Version** and click **Create**.

5.3 Task #3: Add Existing Table

1. Add Contact Table to the solution

- Click to open the **Permit Management** solution you just created.
- Click **Add Existing** and select **Table**.



- Search for **Contact** and select it.
- Click **Next**.
- Click **Select Components**.

1 entities will be added to your project

Contact

No components selected

Select components 

- Select the **Views** tab and select the **Active Contacts** view. Click **Add**.

- Click **Select Components**.
- Select the **Forms** tab and select the **Contact** form.
- Click **Add**.
- You should have **1 View** and **1 Form** selected. Click **Add** again. This will add the Contact Table to the newly created solution.

1 entities will be added to your project

Contact

1 view, 1 form selected

Select components

Add **Cancel**

2. Add User Table to the solution

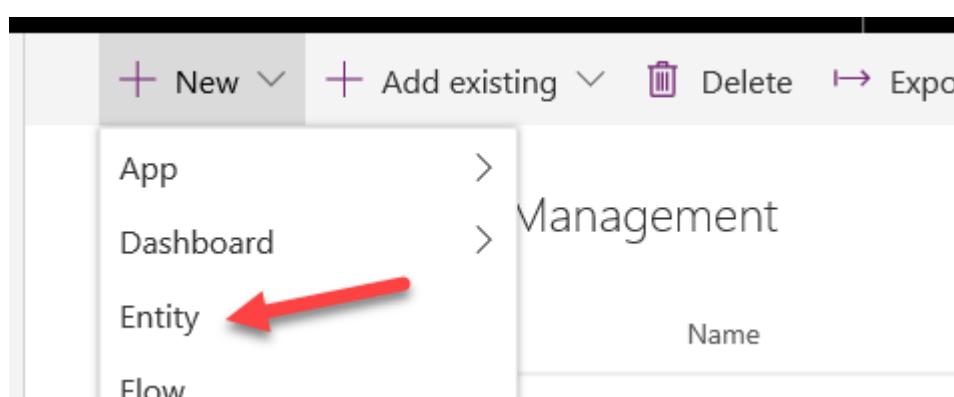
- Click Add Existing and select **Table**.
- Search for **User** and select it.
- Click **Next**.
- **DO NOT** select any components. Click **Add**.
- Your solution should now have two Tables.

6 Exercise #2: Create Tables and Columns

Objective: In this exercise, you will create Tables, add Columns to these Tables and edit the **Status Reason** options for the **Permit** and **Inspection** Tables.

6.1 Task #1: Create Permit Table and Columns

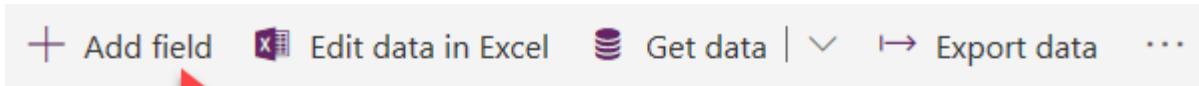
1. Continuing in your development environment, open the Permit Management solution
 - Sign in to [Power Apps maker portal](#)
 - Select **Solutions** and click to open the **Permit Management** solution you just created.
2. Create Permit Table
 - Click **+ New** and select **Table**.



- Enter **Permit** for **Display Name** and click **Create**. This will start provisioning the Table in background while you can start adding Columns.

3. Create Start Date Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.



Solutions > Permit Management > Permit

Fields Relationships Business rules Views Forms Dashboard

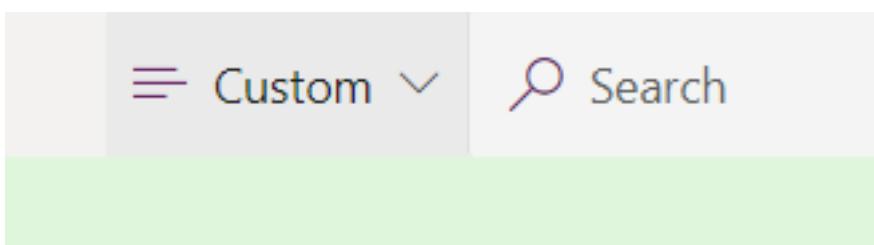
- Enter **Start Date** for **Display Name**.
- Select **Date Only** for **Data Type**.
- Select **Required**.
- Leave the searchable checkbox checked. When a Column is searchable it appears in Advanced Find in model-driven apps and is available when customizing views. De-selecting this will reduce the number of options shown to people using advanced find.
- Click **Done**.

4. Create Expiration Date Column.

- Click **+ Add Column**.
- Enter **Expiration Date** for **Display Name**.
- Select **Date Only** for **Data Type**.
- Click **Done**.

5. Create New Size Column.

- Click **+ Add Column**.
- Enter **New Size** for **Display Name**.
- Select **Whole Number** for **Data Type**.
- Click **Done**.
- Select **Custom** for filter.



- Click **Save Table**.

Fields	Relationships	Business rules	Views	Forms	Dashboards	Charts	Keys	Data
Display name ↑ ↴			Name ↴	Data type ↴	Type ↴	Required ↴	Search... ↴	
Expiration Date	...	contoso_expir...	Date Only	Custom		✓		
Name	...	contoso_name	Text	Custom	✓	✓		
New Size	...	contoso_news...	Whole...	Custom		✓		
Start Date	...	contoso_start...	Date Only	Custom	✓	✓		

Discard **Save Entity**

6.2 Task #2: Create Permit Type Table and Columns

1. Create Permit Type Table

- Click on the solution name. This action will take you back to the Solution.

Fields	Relationships	Business rules	Views	Forms	Dash
Display name ↑ ↴		Name ↴			

- Click + New and select **Table**.
- Enter **Permit Type** for **Display Name**.
- Click **Done**.

2. Create Require Inspections Column

- Make sure you have the **Columns** tab selected and click + **Add Column**.
- Enter **Require Inspections** for **Display Name**.
- Select **Yes/No** for **Data Type**.
- Click **Done**.

3. Create Require Size Column

- Click + **Add Column**.
- Enter **Require Size** for **Display Name**.
- Select **Yes/No** for **Data Type**.
- Click **Done**.

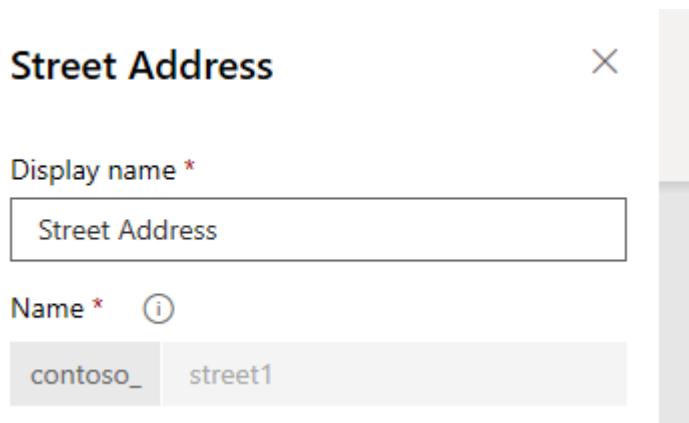
4. Click Save Table

6.3 Task #3: Create Build Site Table and Columns

1. Create Build Site Table

- Click on the solution name. This action will take you back to the Solution.
- Click + New and select **Table**.
- Enter **Build Site** for **Display Name**.
- Change the **Display Name** of the **Primary Column** to **Street Address**.

- Change the **Name** of the **Primary Column** to **street1**.
- Click **Done**.



2. Add City Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **City** for **Display Name** and change the **Name** to **city**.
- Make sure **Text** is selected for **Data Type**.
- Select **Required**.
- Click **Done**.

City X

Display name *

Name * (i)

Data type * (i)

Abc ▼

Required * (i)

▼

Searchable (i)

Calculated or Rollup (i) + Add ▼

Description (i)

Advanced options (i) ▼

Done Cancel

3. Add Zip/Postal Code Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **ZIP/Postal Code** for **Display Name** and change the **Name** to **postalcode**.
- Make sure **Text** is selected for **Data Type**.
- Select **Required**.
- Click **Done**.

4. Add State/Province Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **State/Province** for **Display Name** and change the **Name** to **stateprovince**.

- Make sure **Text** is selected for **Data Type**.
- Select **Required**.
- Click **Done**.

5. Add Country Region Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **Country/Region** for **Display Name** and change the **Name** to **country**.
- Make sure **Text** is selected for **Data Type**.
- Click **Done**.

6. Click **Save Table**.

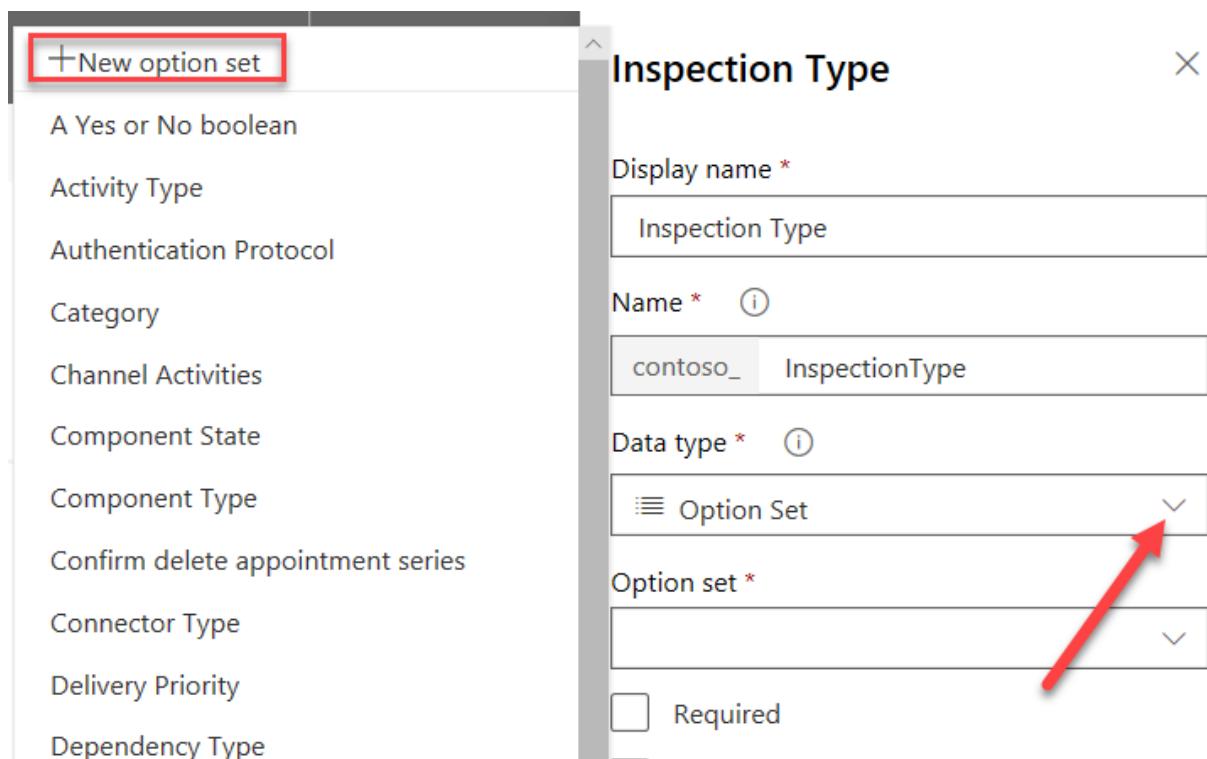
6.4 Task #4: Create Inspection Table and Columns

1. Create Inspection Table

- Click on the solution name. This action will take you back to the Solution.
- Click **New** and select **Table**.
- Enter **Inspection** for **Display Name**.
- Click **Done**.

2. Add Inspection Type Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **Inspection Type** for **Display Name**.
- Select **Choice** for **Data Type**.
- Click on the **Choice** dropdown and select **+New Choice**.



- Enter **Initial Inspection** and click **Add New Item**.

[View more](#)

Items (1)

Initial Inspection

...

Add new item



- Enter Final Inspection and click Save.

Items (2)

Initial Inspection

...

Final Inspection

...

Add new item

Save

Cancel

- Click Done.

3. Add Scheduled Date Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **Scheduled Date** for **Display Name**.
- Select **Date Only** for **Data Type**.
- Select **Required**.
- Click **Done**.

4. Add Comments Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **Comments** for **Display Name**.
- Make sure **Text** is selected for **Data Type**.
- Expand **Advanced options**.
- Set **Max length to 1000** in the Advanced options**.**
- Click **Done**.

5. Add Sequence Column

- Make sure you have the **Columns** tab selected and click **+ Add Column**.
- Enter **Sequence** for **Display Name**.
- Make sure **Text** is selected for **Data Type**.
- Click **Done**.

6. Click **Save Table**.

7. Select **Solutions** on the top and this action will take you back to the Solutions page.

8. Click **Publish All Customizations**.

A screenshot of the Power Apps maker portal's Solutions page. At the top, there is a navigation bar with icons for New solution, Import, Open AppSource, Publish all customizations (which has a red arrow pointing to it), and more. Below the navigation bar is a table titled "Solutions". The table has columns for Display name, Created (sorted by descending date), and Version. There are two rows in the table.

6.5 Task #5: Edit Status Reason Options

1. Open the Permit Management solution

- Navigate to [Power Apps maker portal](#)
- Select **Solutions** from the left menu and click to open the **Permit Management** solution.

2. Switch to Classic

- Click on the ... icon and select **Switch to Classic**.

A screenshot of the Power Apps maker portal's Solutions page. A context menu is open over a solution entry. The menu options are: Clone, Show dependencies, Apply upgrade, and Switch to classic. The "Switch to classic" option is highlighted with a red arrow. The background shows a table of solutions with columns for Name, Type, and Owner.

3. Edit Inspection Table Status Reason options

- Expand **Entities**.
- Expand the **Inspection** Table and select **Fields**.

A screenshot of the Power Apps maker portal's Entities page. On the left, there is a tree view of entities: Components, Entities (Build Site, Contact, Inspection), Forms, Views, Charts, Fields, and Keys. The "Fields" node under "Inspection" is highlighted with a red arrow. On the right, there are details for the "Permit Management" entity: Display Name (Permit Management), Publisher (Contoso), Version (1.0.0.0), and a large Description text area.

- Locate and double click to open the **statuscode** Column.

owninguser	OwningUser	Owning User	Lookup
statecode	statecode	Status	Status
<input checked="" type="checkbox"/> statuscode	statuscode	Status Reason	Status Reason
timezoneruleversionn...	TimeZoneRuleVersion...	Time Zone Ru...	Whole Numbe
utcconversiontimezon...	UTCConversionTimeZ...	UTC Conversi...	Whole Numbe

4. Change the Active option label

- Make sure you have **Active** selected for **Status**.
- Select the **Active** option and click **Edit**.

Type

Data Type *	Status Reason
Status	Active
Active	<input type="button" value="Move Up"/> <input type="button" value="Move Down"/> <input type="button" value="Edit"/> <input type="button" value="Add"/>

- Change the **Label** to **New Request** and click **OK**.

Label *	New Request
Value	1
Color	#0000ff █
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

5. Add the Pending Option

- Click **Add**.

Type

Data Type *	Status Reason
Status	Active
New Request	<input type="button" value="Move Up"/> <input type="button" value="Move Down"/> <input type="button" value="Edit"/> <input type="button" value="Add"/> <input type="button" value="Delete"/>

- Enter **Pending** for **Label** and click **OK**.

6. Add the Passed Option

- Click **Add**.

- Enter **Passed** for **Label** and click **OK**.

7. Add the Failed Option

- Click **Add**.
- Enter **Failed** for **Label** and click **OK**.

8. Add the Canceled Option

- Click **Add**.
- Enter **Canceled** for **Label** and click **OK**.

9. Your option-set should now have 5 options for the **Active** state.

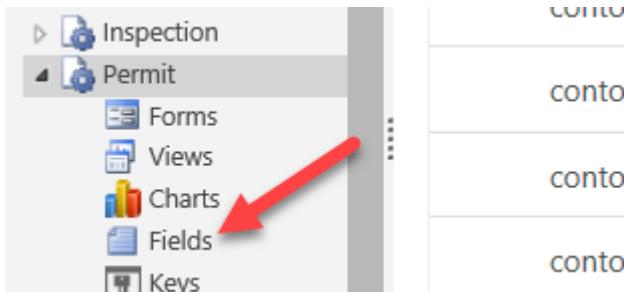
Type	
Data Type *	Status Reason
Status	Active
New Request	
Pending	
Passed	
Failed	
Canceled	

10. Select Pending as the Default Value and click **Save and Close** from the top menu.

		Sort Ascending
		Sort Descending
Default Value		Pending

11. Edit Permit Table Status Reason options

- Expand the **Permit** Table and select **Fields**.



- Locate and double click to open the **statuscode** Column.

12. Add the Locked option

- Make sure you have the **Active** selected for **Status**.
- Click **Add**.
- Enter **Locked** for Label and click **OK**.

13. Add the Completed option

- Click **Add**.
- Enter **Completed** for Label and click **OK**.

14. Add the Canceled option

- Click **Add**.

- Enter **Canceled** for Label and click **OK**.

15. Add the Expired option

- Click **Add**.
- Enter **Expired** for Label and click **OK**.

16. Your option-set should now have 5 options for the **Active** state

For information about how to interact with entities and fields programmatically, see the [Microsoft Dynamics 365 SDK](#)

Type

Data Type *

Status Reason

Status

Active

Active



Locked

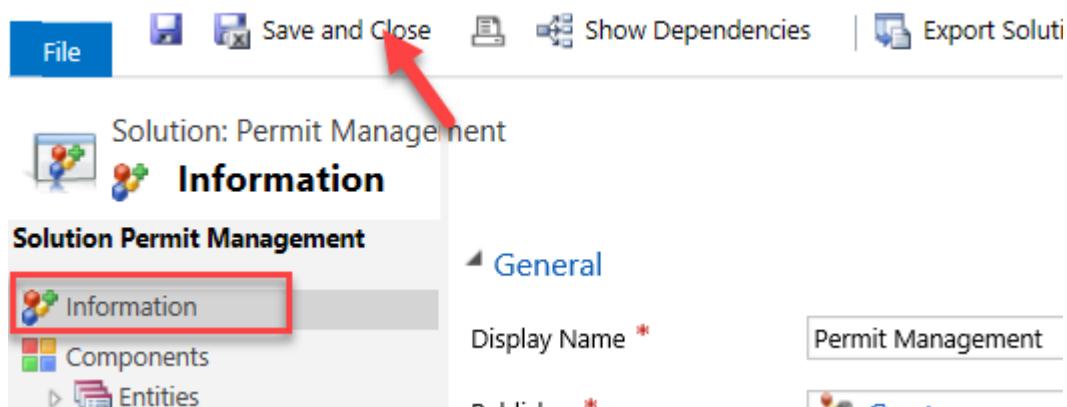
Completed

Cancelled

Expired

17. Select the **Active** for the **Default Value** and click **Save and Close** from the top menu

18. Select **Information** from the left side menu and click **Save and Close** to close classic solution explorer



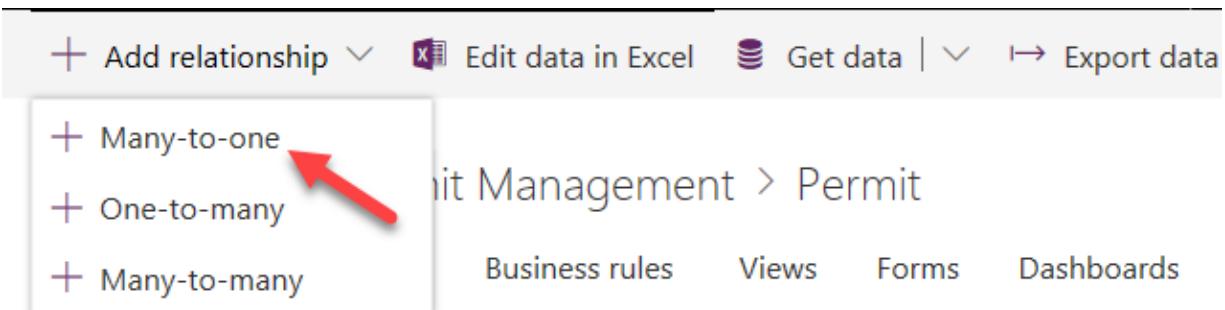
19. Select **Solutions** from the top menu and click **Publish All Customizations**.

7 Exercise #3: Create Relationships

Objective: In this exercise, you will create relationships.

7.1 Task #1: Create Relationships

1. Open the Permit Management solution
 - Sign in to **Power Apps maker portal**
 - Select **Solutions** and click to open the **Permit Management** solution.
2. Create Permit to Contact relationship
 - Click to open the **Permit** Table.
 - Select the **Relationships** tab.
 - Click **+ Add Relationship** and select **Many-to-one**.



- Select Contact for **Related (One)** and click **Done**.

Many-to-one

X

Choose the **Related entity** to which to create your relationship lookup. [Learn more](#)

Current (Many)

Entity *
Permit

Lookup field display name *
Contact

Lookup field name *
contoso_ Contact

Related (One)

* — 1 Entity *
Contact

> General

> Advanced options

3. Create Permit to Inspection relationship

- Click **Add Relationship** and select **One-to-Many**.
- Select **Inspection** for **Table** in the **Related (Many)** and click **Advanced Options**.

Entity *	Entity *
Permit	Inspection
Lookup field display name *	
Permit	
Lookup field name *	
contoso_Permit	

> General

> Advanced options 

- Change the **Type of Behavior** to **Parental** and click **Done**.

Type of behavior * 

Parental

Done

Cancel

4. Create Permit to Build Site relationship

- Click **Add Relationship** and select **Many-to-One**.
- Select **Build Site** for **Related (One) Table** and click **Advanced Options**.
- Change the **Delete** to **Restrict** and click **Done**.

✓ Advanced options

Type of behavior * ⓘ

Referential *

Delete *

Restrict

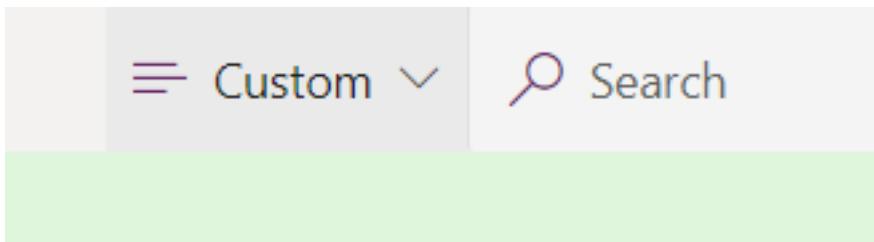
Done

Cancel

5. Create Permit to Permit Type relationship

- Click **Add Relationship** and select **Many-to-One**.
- Select **Permit Type** for **Related (One) Table** and click **Done**.

6. Change the filter to **Custom**.



7. Click **Save Table**.

Fields	<u>Relationships</u>	Business rules	Views	Forms	Dashboards	Charts	Keys	Data
Display name ↑						Relation...	Related...	Relationshi... Type
Build Site						contoso_P...	Build Site	Many-to-one Custom
Contact						contoso_P...	Contact	Many-to-one Custom
Permit						contoso_P...	Inspection	One-to-many Custom
Permit Type						contoso_P...	Permit Type	Many-to-one Custom

Discard

Save Entity

8. Select **Solutions** from the top menu and click **Publish All Customizations**.

8

8.1 lab: title: 'Lab 02: Model driven app'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)

- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

8.2 Lab 02 – Model driven app

9 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab we will continue to build on top of the components created in the previous Lab. We will now build a Power Apps model-driven app to allow the office staff manage records for the inspectors and the inspectors to manage their own records as needed.

10 High-level lab steps

As part of creating the model-driven app, you will complete the following:

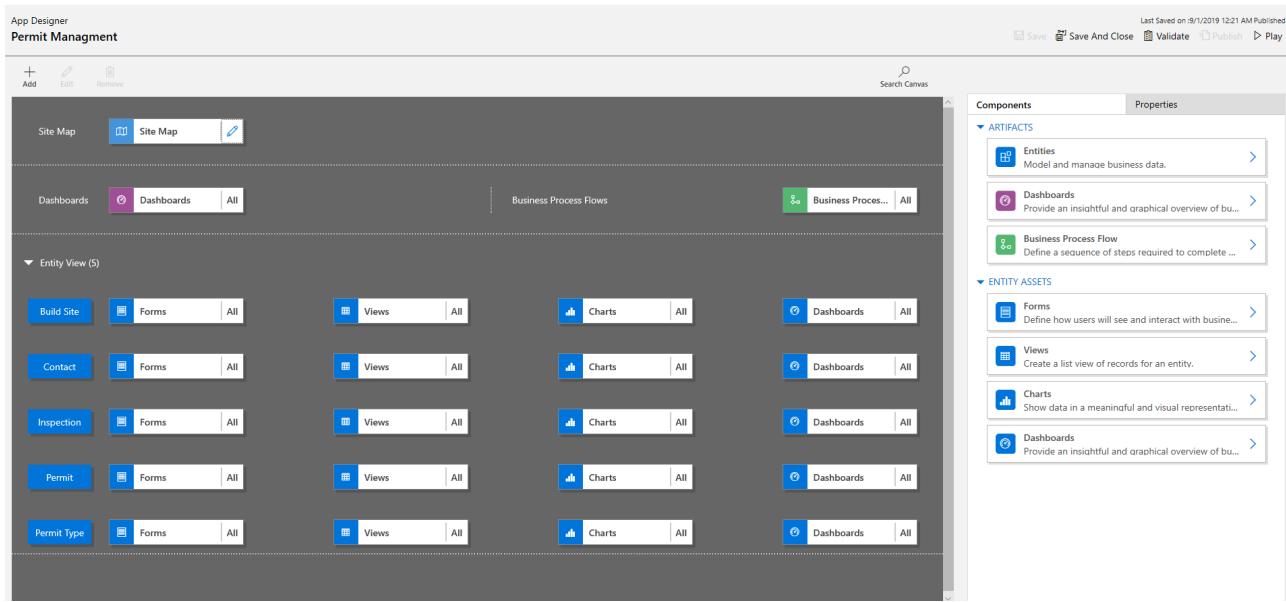
- Create a new model-driven app named Permit Management
- Edit the app navigation to reference the required Tables
- Customize the forms and views of the required Tables for the app

Views: As the name suggests, this helps viewing the existing data in the form of table. This is the configuration of the Columns that will be displayed on the screen.

Forms: This is where the user creates/updates new records in the Tables.

Both will be integrated to the model-driven app for a better user-experience.

The following is what the model-driven app designer looks like when all the customizations are completed:



10.1 Things to consider before you begin

- What changes should we make to improve the user experience?
- What should we include in a model-driven app based on the data model we've built?
- What customizations can be made on the sitemap of a model-driven app?
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production once everything is built and tested.

11 Exercise #1: Customize Views and Forms

Objective: In this exercise, you will customize views and forms of the custom created Tables that will be used in the model-driven app.

11.1 Task #1: Edit Permit Form and View

1. In your development environment, open the Permit Management solution.

- Sign in to [Power Apps maker portal](#)
- Select your **Dev environment**.
- Select **Solutions**.
- Click to open the **Permit Management** solution.

2. Steps to edit the Permit Table form.

- Click to open the **Permit** Table.
- Select the **Forms** tab and click to open the **Main** form. By default, the form has two Columns, Name (Primary Column) and Owner.

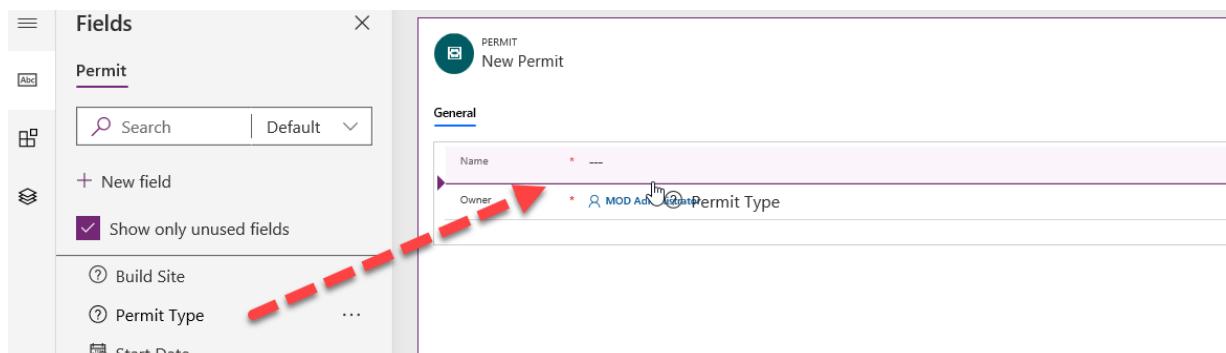
Solutions > Permit Management > Permit

Fields Relationships Business rules Views **Forms** Dashboards C

 Model-driven

Name ↑ ↓	Form type	Type
Information	... Main	Custom
Information	... Quick View Fo...	Custom
Information	... Card	Custom

- Drag the **Permit Type** Column to the form and place it below the **Name** Column.



The screenshot shows the 'Fields' list for the 'Permit' entity on the left and the Power Apps canvas on the right. The 'Fields' list includes 'Name' (Primary), 'Owner', 'Build Site', 'Permit Type', and 'Start Date'. The Power Apps canvas displays a 'New Permit' form with fields for 'Name', 'Owner', and 'Permit Type'. A red dashed arrow indicates the drag operation from the 'Permit Type' field in the list to its position on the canvas form.

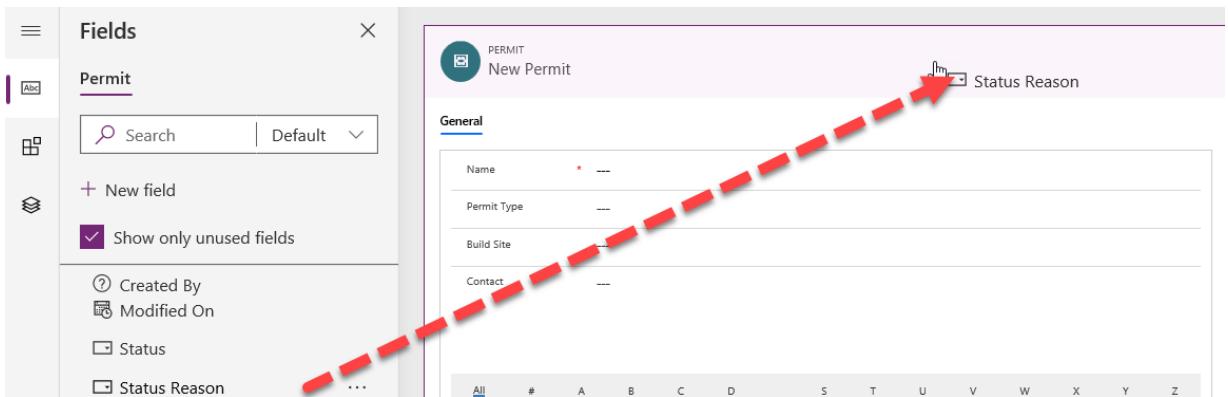
New Permit

Permit

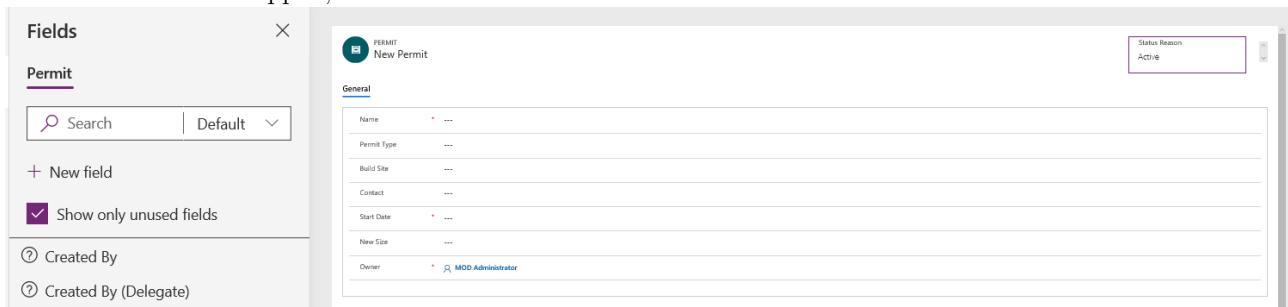
General

Name	*	---
Permit Type	---	
Build Site	---	
Contact	---	
Start Date	*	---
New Size	---	
Owner	*	

- Add Build Site lookup, Contact lookup, Start Date and New Size to the form.
- Drag the **Status Reason** Column and drop it in the right side of the form header.

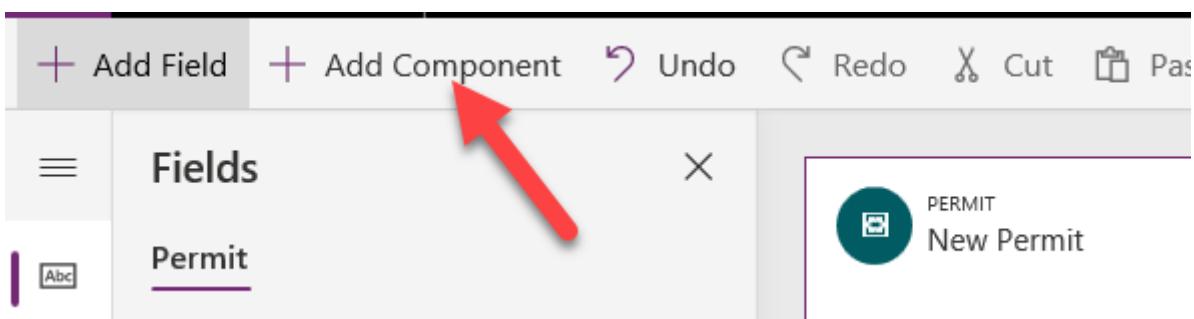


Once the control is dropped, this form will look like:

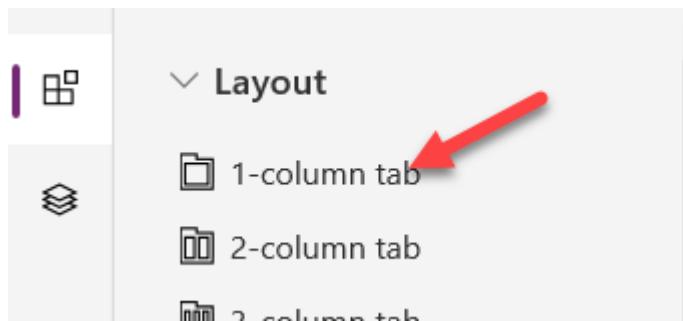


3. Add new tab for **Inspections** to the form.

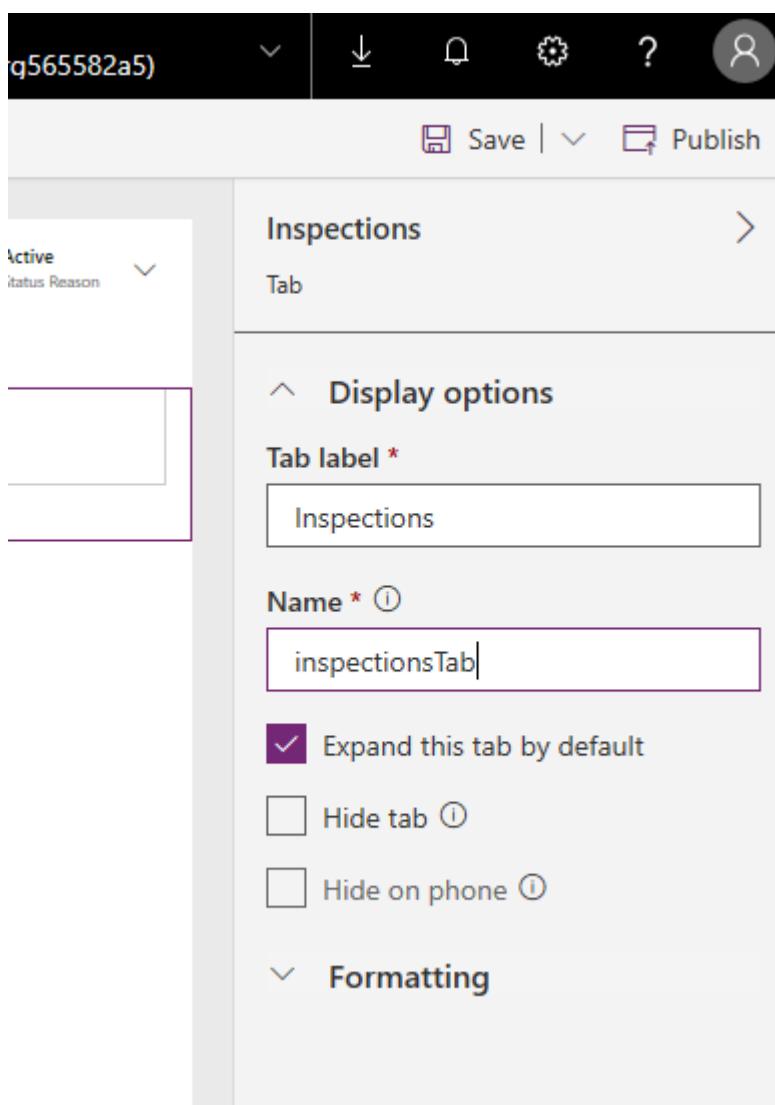
- With focus set on the main body of the form (not in the header) click **Add Component**.



- Select **One Column Tab**.

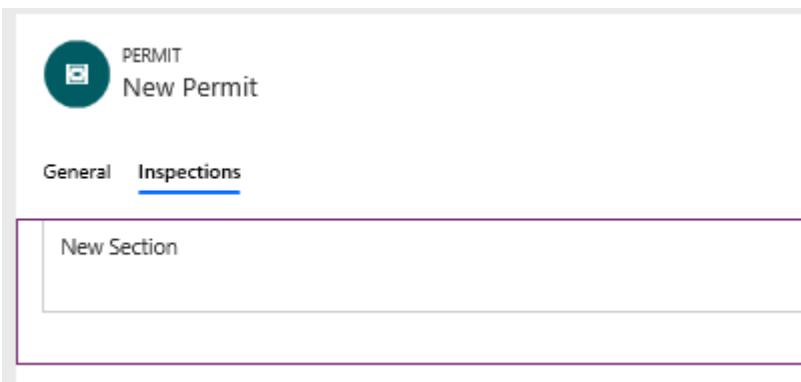


- Select the new tab you added.
- Go to the **Properties** pane, change the **Tab Label** to **Inspections** and the **Name** to **inspectionsTab**.



4. Steps to add Sub-Grid to the Permit form.

- Select the **Inspections** tab. Make sure that you have selected the whole Tab and not just a section.



- Click **Add Component**.
- Scroll down and select **Subgrid**, this will open a pop-up to select Table.
- Check the **Show Related Records** checkbox, select **Inspections** for **Table**, select **Active Inspections** for **Default View** and click **Done**.

The image consists of two side-by-side screenshots. The left screenshot shows a sidebar titled 'Add Component' with various options like Arc Knob, Radial Knob, Flip - Switch, Number Input, Display, Website Preview, Related data, Quick view, and Subgrid. The 'Subgrid' option is highlighted with a red box. The right screenshot shows a modal dialog titled 'Select subgrid views' with fields for 'Entities *' (set to 'Inspections (Permit)') and 'Default View *' (set to 'Active Inspections'). Both dialogs have 'Done' and 'Cancel' buttons at the bottom.

5. Edit Sub-Grid properties.
 - Go to the sub-grid properties pane and change the Label to **Inspections**.

Save | Publish

Inspections

Subgrid

Display options

Label *

Inspections

Name

6. Steps to hide the section label

- Click to select the section.

PERMIT

New Permit

General Inspections

New Section

Inspections

✓ Name ↑ | Inspection Type

- Go to the **Properties** pane and check the **Hide Label** checkbox.

Display options

Section label *

New Section

Name * ⓘ

tab_2_section_1

Hide label

Lock section ⓘ

7. Click **Save** and wait for the save to complete.

8. Click **Publish** and wait for the publishing to complete.

9. Click on the <- Back button. You should now be back to the Permit Table Forms tab.

10. Steps to edit the Active Permits view.

- Select the **Views** tab and click to open the **Active Permits** view.

Solutions > Permit Management > Permit

Fields Relationships Business rules **Views** Forms Dashboards Charts Keys Data

The screenshot shows the 'Views' tab selected in the top navigation bar. Below it, there are two views listed: 'Active Permits' and 'Inactive Permits'. A red arrow points to the 'Active Permits' view. On the right side of each view entry, there is a 'View type' dropdown set to 'Public View'.

- Drag the **Build Site** Column and drop it between the **Name** and **Created On** Columns.

The screenshot shows the 'Fields' pane on the left with the 'Permit' tab selected. In the center, the view columns are listed: Name ↑, Created On ↓, and Build Site. A dashed red arrow indicates the movement of the 'Build Site' column from its original position to a position between 'Name' and 'Created On'. A circular icon with a grid symbol is visible in the center of the view area.

- Click on the **Permit Type** Column. The Permit Type Column will be added to the view.
- Click on the **Contact** Column. The Contact Column will be added to the view.
- Go to the view designer and click on the chevron icon of the **Created On** column.

The screenshot shows the view columns again: Name ↑, Build Site ↓, Created On ↓, Permit Type ↓, and Contact ↓. A red arrow points to the chevron icon on the 'Created On' column header.

- Click **Remove**. **Created On** Column will now be removed from the view.
- Click **Save** and wait until the changes are saved.
- Click **Publish** and wait for the publishing to complete.

11. Click on the <-Back button.

11.2 Task #2: Edit Build Site Form and View

1. Open the Permit Management solution.

- Sign in to Power Apps maker portal
- While in your dev environment, select **Solutions**, and click to open the **Permit Management** solution.

2. Edit the Build Site Table form.

- Click to open the **Build Site** Table.
- Select the **Forms** tab and click to open the **Main** form.
- Add **City**, **State/Province**, **Zip/Postal Code**, and **Country Region** Columns to the form between **Street Address** and **Owner**.

New Build Site

Build Site

General

Street Address	*	---
City	*	---
State/Province	*	---
ZIP/Postal Code	*	---
Country/Region	---	
Owner	*	 MOD Administrator

3. Click **Save** and wait until the changes are saved.
4. Click **Publish** and wait for the publishing to complete.
5. Click on the <-Back button.
6. Edit the Active Build Sites view.
 - Select the **Views** tab and click to open the **Active Build Sites** view.
 - Add **City** and **Zip/Postal Code** to the view.
 - Remove **Created On** from the view by selecting **Remove** from the options in Column Chevron.

Street Address ↑ \v	City \v	ZIP/Postal Co... \v	 Add column
			

7. Click **Save** and wait until the changes are saved.
8. Click **Publish** and wait for the publishing to complete.
9. Click on the <-Back button.

11.3 Task #3: Edit Inspection Form and View

1. Open the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)
 - While in your dev environment, select **Solutions** and click to open the **Permit Management** solution.
2. Edit the Inspection Table form.
 - Click to open the **Inspection** Table.

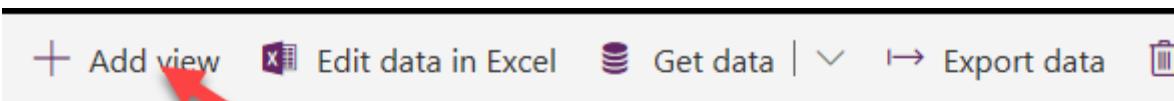
- Select the **Forms** tab and click to open the **Main** form.
- Add **Inspection type**, **Permit**, **Scheduled Date**, and **Comments** Columns to the form. **Inspection type**, **Permit**, **Scheduled Date** should be added between **Name** and **Owner**, while **Comments** will be added after the **Owner** Column.
- Drag the **Status Reason** Column **Comments** and drop it in the right side of the form header.

- The form should now look like the image below.

3. Click **Save** and wait until the changes are saved.
4. Click **Publish** and wait for the publishing to complete.
5. Click on the <-Back button.
6. Edit the Active Inspections view.
 - Select the **Views** tab and click to open the **Active Inspections Sites** view.
 - Add **Inspection Type**, **Scheduled Date** and **Sequence** to the view.
 - Remove **Created On** from the view by selecting the chevron on the Column and select **Remove**.

Name ↑ ↴	Inspection Type ↴	Scheduled Date ↴	Sequence ↴

7. Click **Save** and wait until the changes are saved.
8. Click **Publish** and wait for the publishing to complete.
9. Click on the <-Back button.
10. Create new Inspector View for the Inspection Table.
 - Make sure you still have the **Views** tab selected.
 - Click **+ Add View**. This will open a new window to create View.



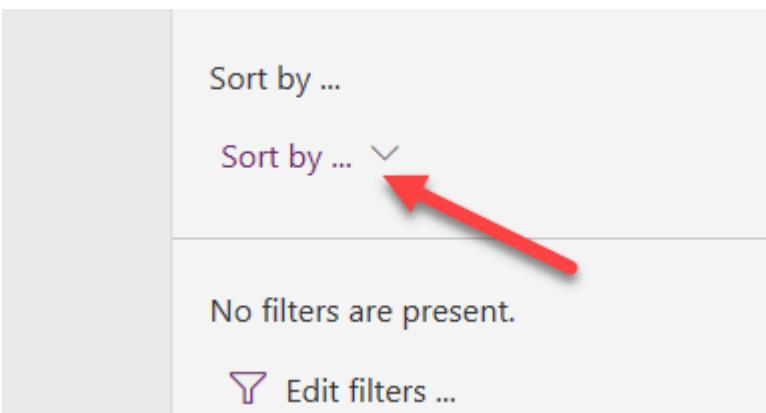
Solutions > Permit Management > Inspection

Fields Relationships Business rules Views Forms Dashboard

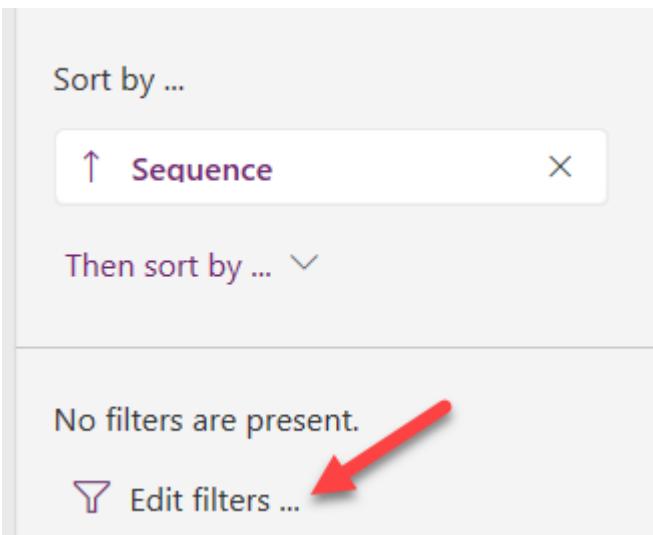
- Enter **Inspector View** for **Name** and click **Create**.
- Add **Inspection Type**, **Permit**, **Scheduled Date**, and **Sequence** Columns to the view.

Name ↴	Inspection Type ↴	Permit ↴	Scheduled Date ↴	Sequence ↴

11. Sort the Inspector View by the sequence.
 - Go to the view properties pane and click **Sort By**.

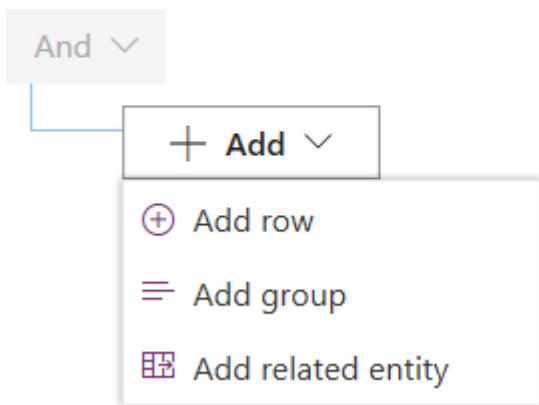


- Select **Sequence**.
12. Filter the Inspector View.
 - Go to the view properties pane and click **Edit Filter**. This will open a new pop-up on the right side of the window.

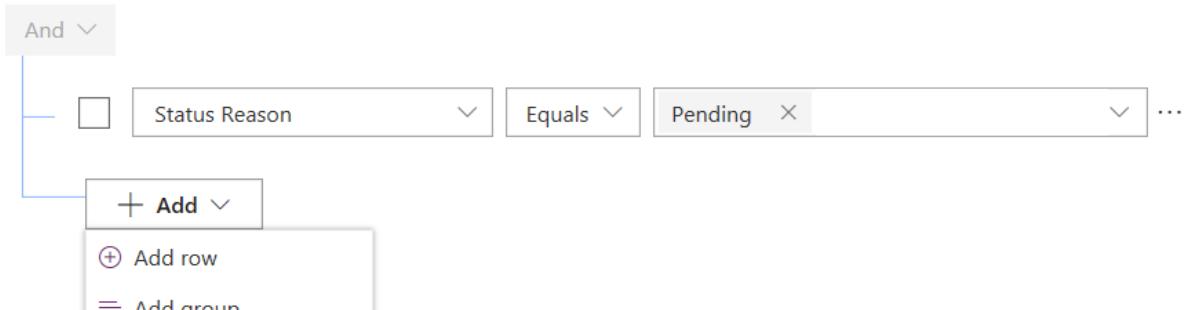


- Click **Add** and select **Add Row**.

Edit filters



- Set the filter property by Selecting **Status Reason** in first dropdown and **Pending** in the third dropdown**. Now, click **Add** and select **Add Row** again.



- To set the filter property select **Owner** Column in the first dropdown and Equals current user in second dropdown and click **OK**.

Edit filters

```
graph TD; And[And] --> StatusReason[Status Reason Equals Pending]; And --> Owner[Owner Equals current user ...]; Add[+ Add]
```

13. Click **Save** and wait until the changes are saved.
14. Click **Publish** and wait for the publishing to complete.
15. Once the changes are published, close the window and click **Done** on the previous window for the Power Apps.

11.4 Task #4: Edit Permit Type Form

1. Open the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)
 - Select your **Dev environment**.
 - Select **Solutions**.
 - Click to open the **Permit Management** solution.
2. Edit the Permit Type Table form.
 - Click to open the **Permit Type** Table.
 - Select the **Forms** tab and click to open the **Main** form.
 - Add **Require Inspections** and **Require Size** Columns to the form between **Name** and **Owner**.

General	
Name	*
Require Inspections	---
Require Size	---
Owner	*

- Click **Save** and wait until the changes are saved.
3. Click **Publish** and wait for the publishing to complete.
 - Click on the <-Back button.
 4. Edit the **Permit Type** Table **Active Permit Type** View.

- Select the **Views** tab and click to open the **Active Permit Type** view.
- Add **Require Inspections** and **Require Size** to the view.
- Remove **Created On** from the view but selecting the chevron on the Column and select **Remove**.

Name ↑ ↓	Require Inspe...	Require Size

- Click **Save** and wait until the changes are saved.
- Click **Publish** and wait for the publishing to complete.
- Click on the <-Back button.

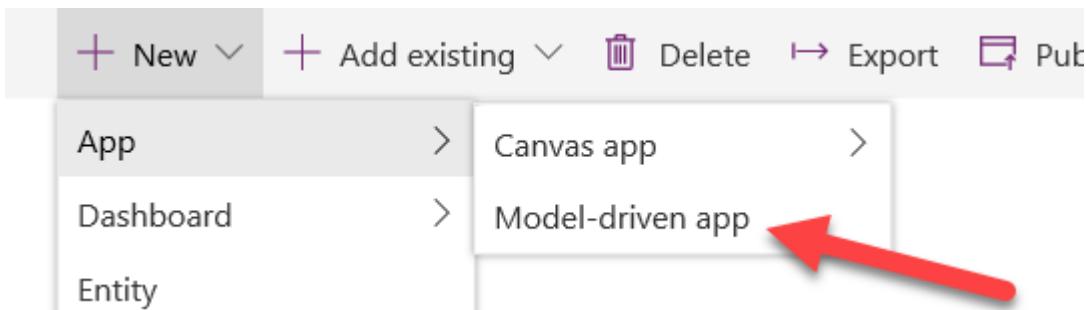
12 Exercise #2: Create Model-Driven Application

Objective: In this exercise, you will create the model-driven app, customize the sitemap, and test the app.

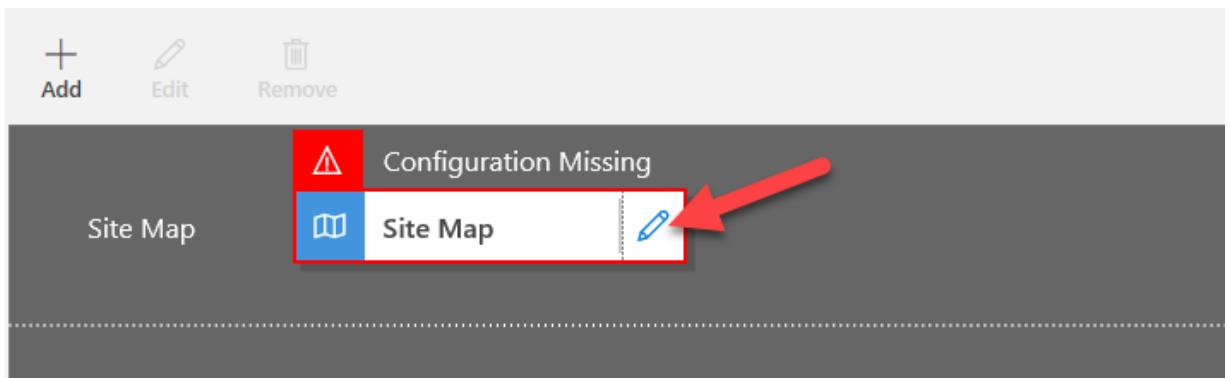
Note: You will see several Columns not addressed as you build out your application, particularly on the sitemap steps. We have taken some short cuts in the interest of time for doing the labs. In a real project you would give these items logical names.

12.1 Task #1: Create Application

1. Open the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)
 - While in your dev environment, click to open the **Permit Management** solution.
2. Create the Model-Driven Application
 - Click **New** and select **App | Model-Driven App**. This will open a new Window.



- Enter **Permit Management** for **Name** and click **Done**.
- 3. Edit Sitemap
 - Click **Edit Site map**.



4. Edit the default titles

- Select **New Area**.
- Go to the properties pane and enter **Building Dept.** for **Title**.
- Select **New Group**.
- Go to the **Properties** pane and enter **Permits** for **Title**.

A screenshot of a software interface showing the properties of a selected node. At the top, there is a toolbar with several icons: 'Add' (plus sign), 'Cut' (scissors), 'Copy' (copy), 'Paste' (paste), 'Clone' (clone), and 'Delete' (trash can). Below the toolbar, the node's title is displayed as 'Building Dept.' in bold blue text. Underneath the title, there is a horizontal line separator. Below the line, the word 'Permits' is shown in blue text. Further down, the text 'New Subarea' is visible. A red arrow points from the bottom right towards the 'Edit' icon in the toolbar.

5. Add the Permit Table to the sitemap

- Select **New Subarea**.
- Go to the **Properties** pane and select **Table** from the dropdown for **Type**.
- Search for **Permit** Table from the dropdown for **Table**.

SUB AREA

General

Type

Entity

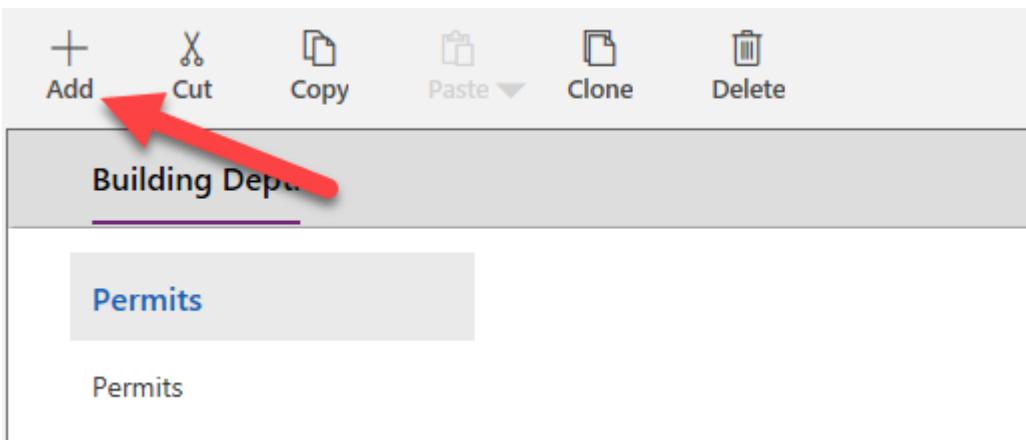
Entity *

Permit

URL

6. Add the Inspection Table to the sitemap

- Select **Permits** group and click **Add**.



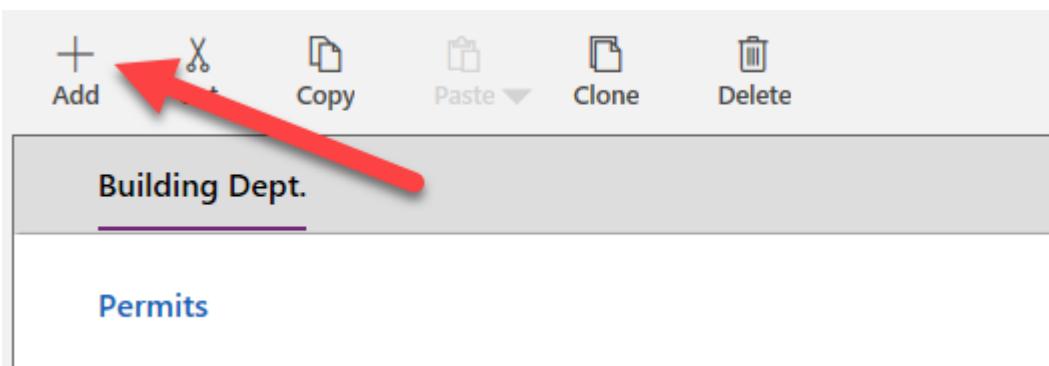
- Select **Subarea**.
- Go to the **Properties** pane.
- Select **Table** from the dropdown for **Type** and search for **Inspection Table** from the dropdown for **Table**.

7. Add the Permit Type Table to the sitemap

- Select **Permits** group and click **Add**.
- Select **Subarea**.
- Go to the **Properties** pane.
- Select **Table** from the dropdown for **Type** and search for **Permit Type** Table from the dropdown for **Table**.

8. Add new Group to the sitemap

- Select the **Building Dept.** area and click **Add**.



- Click **Add** and select **Group**.
- Select **Group**.
- Go to the **Properties** pane and enter **Contacts** for Title.

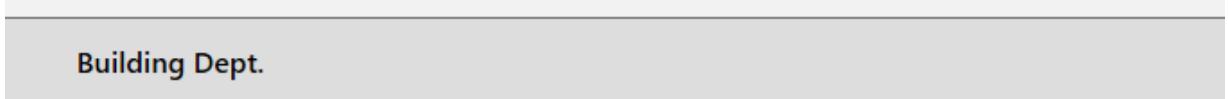
9. Add the Contact Table to the Contacts group.

- Select the **Contacts** group.
- Click **Add** and select **Subarea**.
- Go to the **Properties** pane.
- Select **Table** from the dropdown for **Type** and search for **Contact** Table in the dropdown for **Table**.

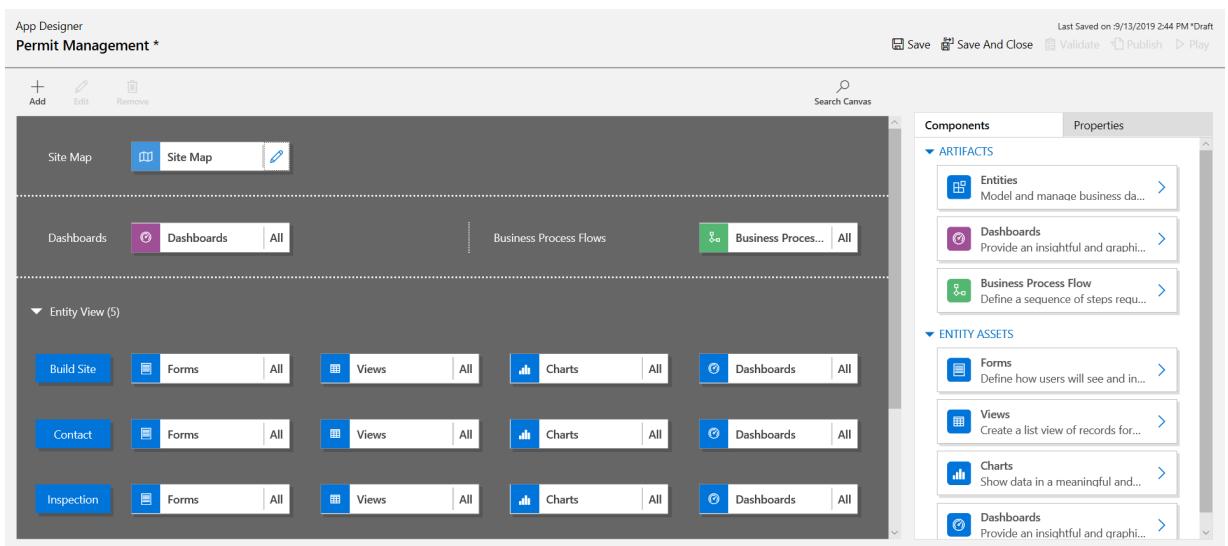
10. Add the Build Site Table to the Contacts group.

- Select the **Contacts** group.
- Click **Add** and select **Subarea**.
- Go to the **Properties** pane.

- Select **Table** from the dropdown for **Type** and search for **Build Site Table** in the dropdown for **Table**.
11. The sitemap should now look like the image below.



12. Click **Save**, this will show the loading screen while the changes are getting saved.
13. Click **Publish** to publish the sitemap and wait for the publishing to complete.
14. Click **Save and Close** to close the sitemap editor.
15. You will see the assets for the Tables that were added to the sitemap are now all in the application.



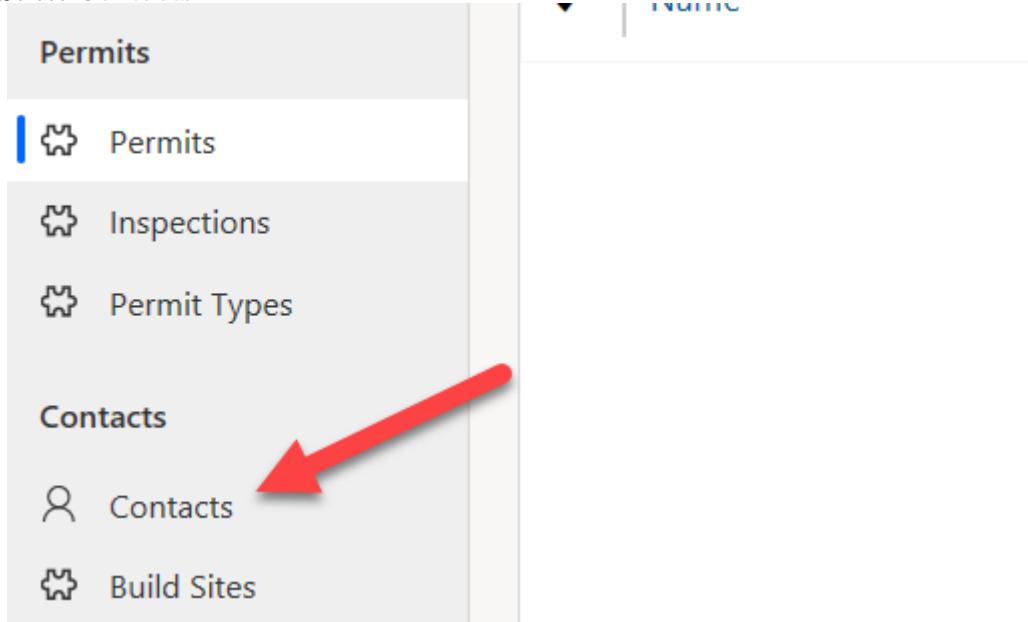
16. Click **Save** to save the application.
17. Click **Validate** to validate the changes done in the application. This will show some warnings. Feel free to review them, but we can ignore them, since we have not referenced a specific View and Form for the Tables and this will display all the Views and Forms.
18. Click **Publish** to publish the application and wait for the publishing to complete.
19. Click **Save and Close** to close the app designer.
20. Click **Done**.
21. Click **Publish all Customizations**.
22. Select **Apps** and your application should now be listed in the Recent Apps.

Apps in Rage Dev

□	Name	Modified	Owner
■	Permit Management	... 44 sec ago	MOD /

12.2 Task #2: Test Application

1. Start the application
 - Select **Apps** and click to open the **Permit Management** app in a new window.
2. Create new Contact record
 - Select **Contacts**



- Click **New** from the top menu.
- Provide **First Name** as **John** and **Last Name** as **Doe**.
- Click **Save and Close**.

CONTACT New Contact

Summary Details

CONTACT INFORMATION

First Name * John

Last Name * Doe

- You should now see the created contact on the **Active Contacts** view.

My Active Contacts ▾

Full Name	Email
John Doe	---

3. Create new Build Site record

- Select **Build Sites** from the sitemap.
- Click **New**.
- Provide **Street Address**, **City**, **State/Province**, **Zip/Postal Code**, and **Country/Region** as:
Street Address: One Microsoft Way

City: Redmond

State/Province: WA

ZIP/Postal Code: 98052

Country/Region: USA

- Click **Save and Close** and this will show the newly created record on the Active Build Sites View.

Active Build Sites ▾

✓ Street Address

One Microsoft Way

4. Create new Permit Type record

- Select **Permit Types** from the sitemap.
- Click **New**.
- Provide **Name** as **New Construction** and click **Save and Close**. This will create the record and you should be able to see it on the Active Permit Type View.

Active Permit Types

 Name

New Construction

5. Create new Permit record

- Select **Permits** from the sitemap.
- Click **New**.
- Provide **Name** as **Test Permit**, select the **Permit Type**, **Build Site**, and the **Contact** records you created in the previous steps.
- Select a future date for the **Start Date** and click **Save**.

 Save  Save & Close  New  Flow 



PERMIT

New Permit

General Inspections

Name

* **Test Permit**

Permit Type

 New Construction

Build Site

 One Microsoft Way

Contact

 John Doe

Start Date

* 8/30/2019

New Size

Owner

*  MOD Administrator

6. Create new Inspection record

- Go to the **Inspections** tab.
- Click **+ New Inspections**.

Test Permit
Permit

General Inspections Related

Active
Status Reason ▾

+ New Inspection ...

Search for records

Name	Inspection Type	Scheduled Date	Sequence

- Provide Name as **Framing Inspections**, select **Initial Inspection** from the dropdown for **Inspection Type**, and select future date for **Scheduled Date**.
- Click **Save and Close**.

Save Save & Close New Flow ▾

INSPECTION

New Inspection

General

Name * **Framing Inspection**

Inspection Type **Initial Inspection**

Permit **Test Permit**

Scheduled Date * **8/30/2019**

Owner * **MOD Administrator**

- The **Inspection** record should now show on the **Permit** sub-grid.

Test Permit
Permit

General Inspections Related

Active
Status Reason ▾

+ New Inspection ...

Search for records

Name	Inspection Type	Scheduled Date	Sequence
Framing Inspections	Initial Inspection	12/27/2019	---

All # A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

- You may add more test records.

12.3 lab: title: 'Lab 03: Canvas app'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

12.4 Lab 03 – Canvas app

13 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

As we continue to build our solution, we will now design a Power Apps canvas app that the inspectors will use in the Column on their mobile devices. Canvas apps are low code apps that can be designed for a tablet or mobile phone layouts. You will build a two-screen canvas app that allows inspectors quickly access and process the inspections.

14 High-level lab steps

We will follow the below schema to design the canvas app:

- Create the app using the tablet form factor
- Connect to Microsoft Dataverse as a data source
- Configure a gallery control to show the pending inspections
- Use a Microsoft Dataverse view to populate the gallery
- Configure a detail page with inspection info
- Handle saving the inspection results to Microsoft Dataverse
- Export the solution with the data model and apps and import it to the “Production” environment

This is the first screen in the application to show all Pending Inspections for the logged in Inspector.

My Pending Inspections	
Final inspection	>
9/19/2019	
Smith inspection	>
9/24/2019	
Duplex on Main	>
10/3/2019	

This second screen lets the inspector update the selected Inspection.

Duplex on Main

Name Duplex on Main	Scheduled Date 10/3/2019	Inspection Result <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> Pending X New Request Pending Passed Failed Inactive </div>
Comments		
<input type="button" value="Submit"/>		

14.1 Things to consider before you begin

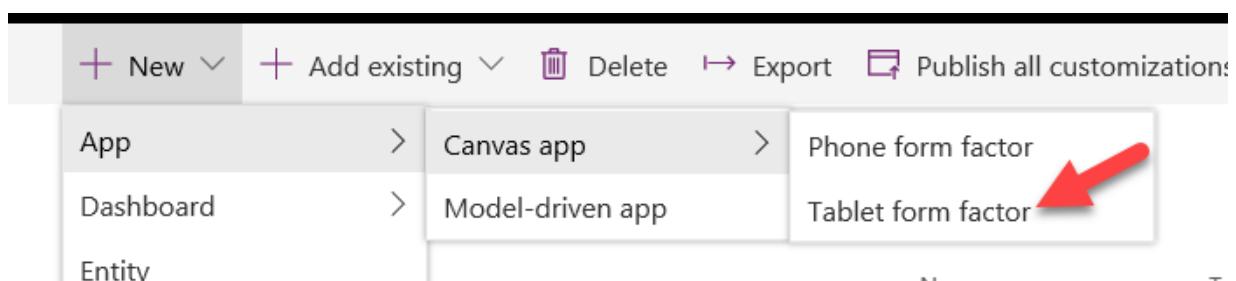
- What would an inspector need quick access to while in the field?
- How do we move our solution from the development to the production environment?
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production in Exercise 2 of this lab.

15 Exercise #1: Create Canvas App

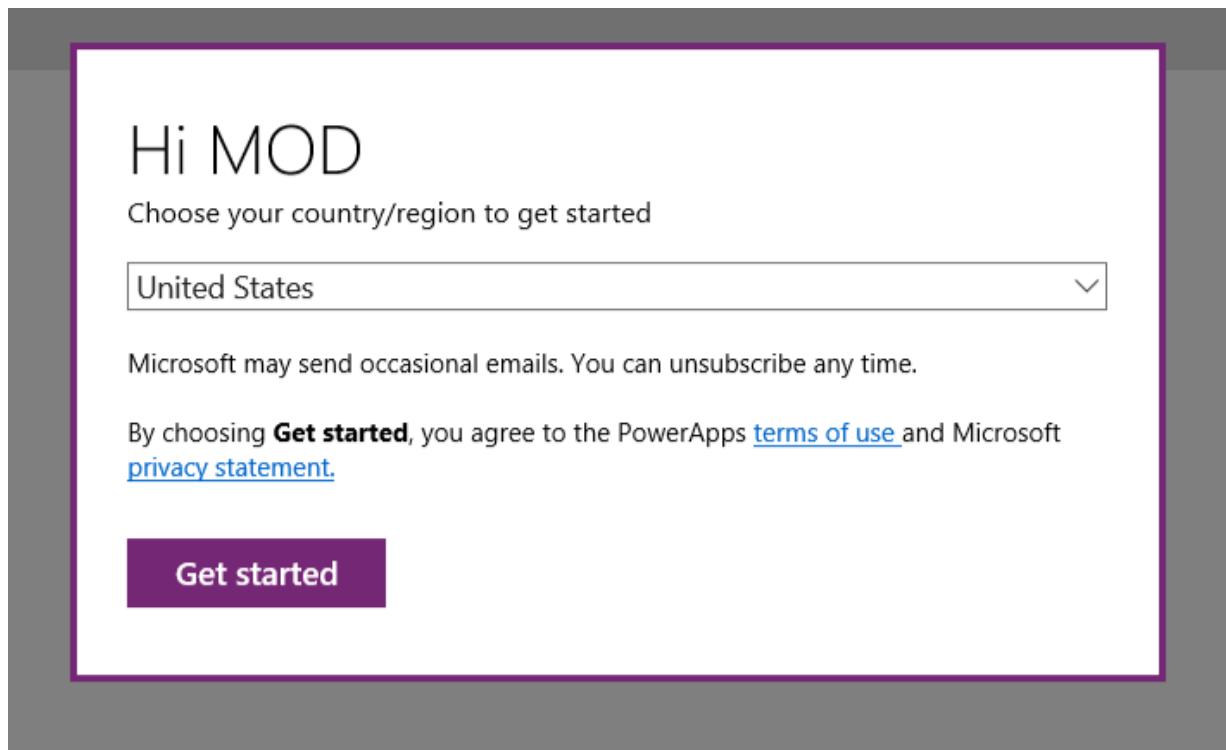
Objective: In this exercise, you will create a canvas app.

15.1 Task #1: Create Canvas App

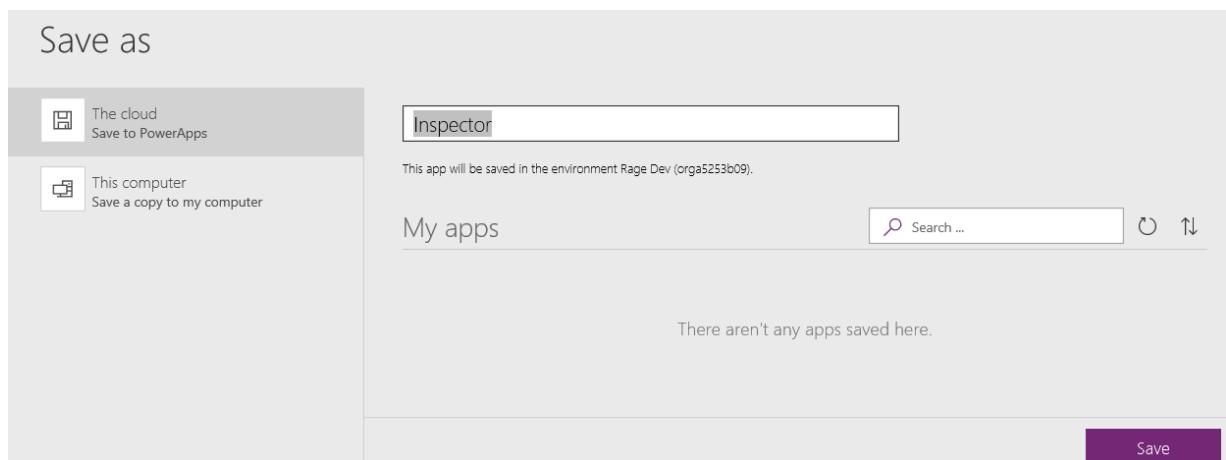
1. Open the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)
 - Select your **Dev environment**.
 - Select **Solutions**.
 - Click to open the **Permit Management** solution.
2. Create new canvas application
 - Click **+ New** and select **App | Canvas App | Tablet Form Factor**. This will open the App Editor in a New window.



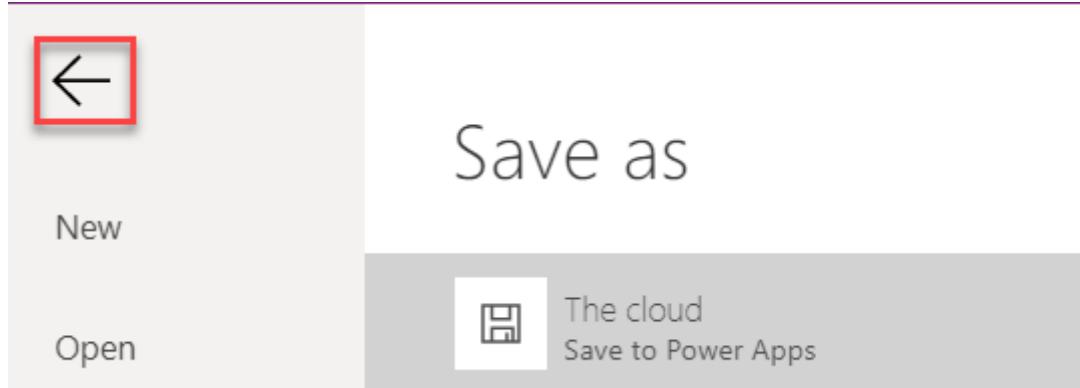
If you are creating your first app, this will ask you to set the Country/region for the app. Click **Get Started**.



- Click **File** and select **Save As**.
- Select **The Cloud**, enter **Inspector** for Name, and click **Save**. This will make sure that the changes are not removed if the app closes unexpectedly.

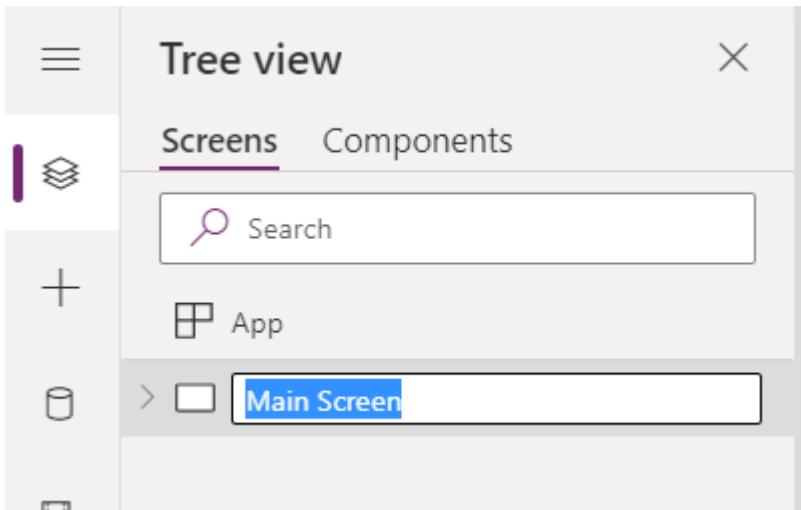


- Click on the app designer button.



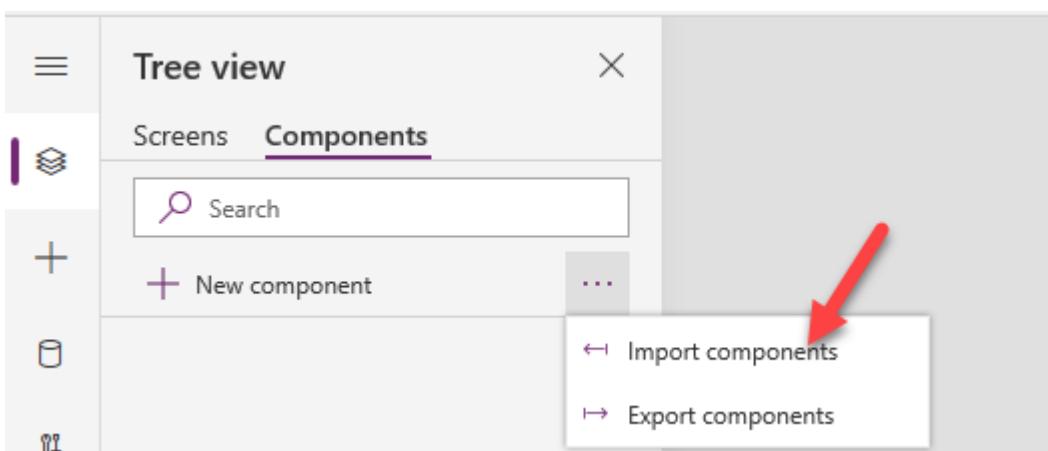
3. Rename Screen1.

- Go to the **Tree View** and double click on **Screen1**.
- Rename it **Main Screen** and press **Enter**.



4. Import Component.

- Select the **Components** tab.
- Click on the ... (Component Options) button and select Import Components.



- Click **Upload File**.
- Browser to the lab resources folder, select the **Components** file and click **Open**.

Name: Components.msapp

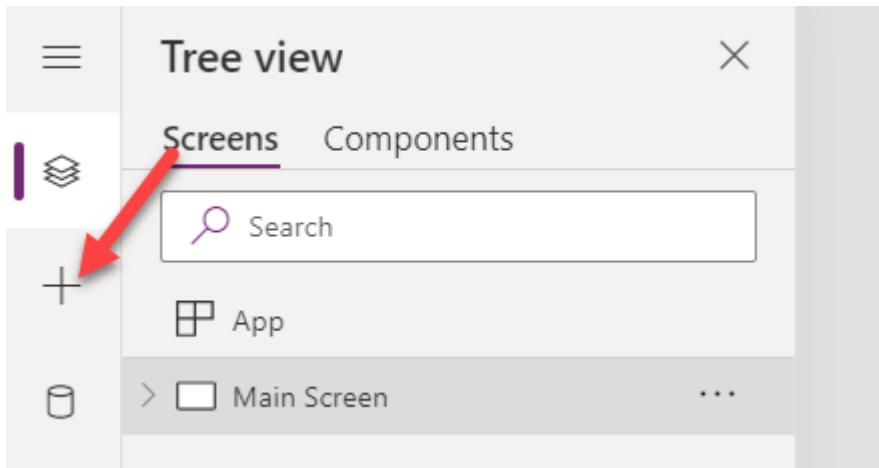
Type: All files (*)

Size: 29 KB

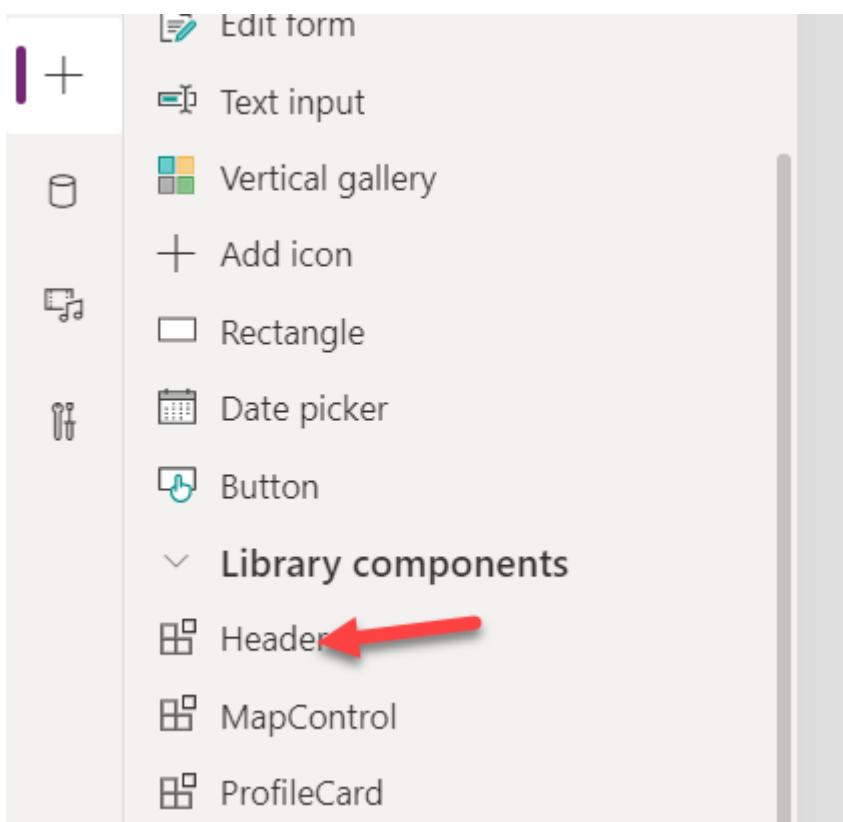
Open Cancel

5. Add the Header component to the MainScreen.

- Select the **Screens** tab.
- Click **+** Insert.



- Click to expand **Library components** and select **Header**.



- Select **Tree View**.
- Rename **Header_1** to **Main Header** by double click on Header_1.

6. Change the Main Header properties

- Select **Main Header**.
- Change the **Text Column** of the **Main Header** to **My Pending Inspections**. This can be done by selecting “Text” property in the dropdown below top menu. Make sure that you have selected the Main Header control while doing this step.

The screenshot shows the Power Apps formula bar. On the left, there is a dropdown menu set to "Text". To its right is an equals sign (=) followed by a formula bar containing "My Pending Inspections" in red text. Below this, a tooltip displays the formula: "'My Pending Inspections'" = My Pending Inspections. A preview of the screen shows a blue header bar with the text "My Pending Inspections".

- Change the **Width** value of the **Main Header** to the formula below.

`Parent.Width`

The screenshot shows the Power Apps formula bar. The left section shows "Width" in a dropdown. To its right is an equals sign (=) followed by a formula bar containing "Parent.Width". Below this, a tooltip displays the formula: Parent.Width = 1. A note at the bottom says "and to show names".

15.2 Task #2: Add Data Source

1. Add Permit and Inspection as data source.

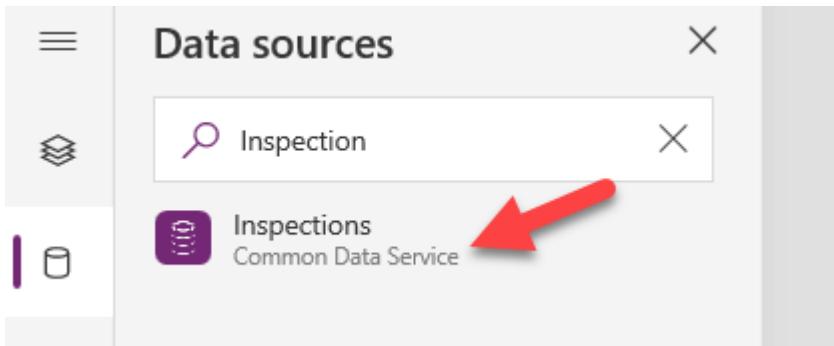
- Select the **Data Sources** tab.

The screenshot shows the "Tree view" screen with the "Screens" tab selected. In the left sidebar, there is a "Data Sources" icon with a red arrow pointing to it. The main area shows a search bar and a list of components: "App", "Main Screen" (which is expanded), and "Main Header".

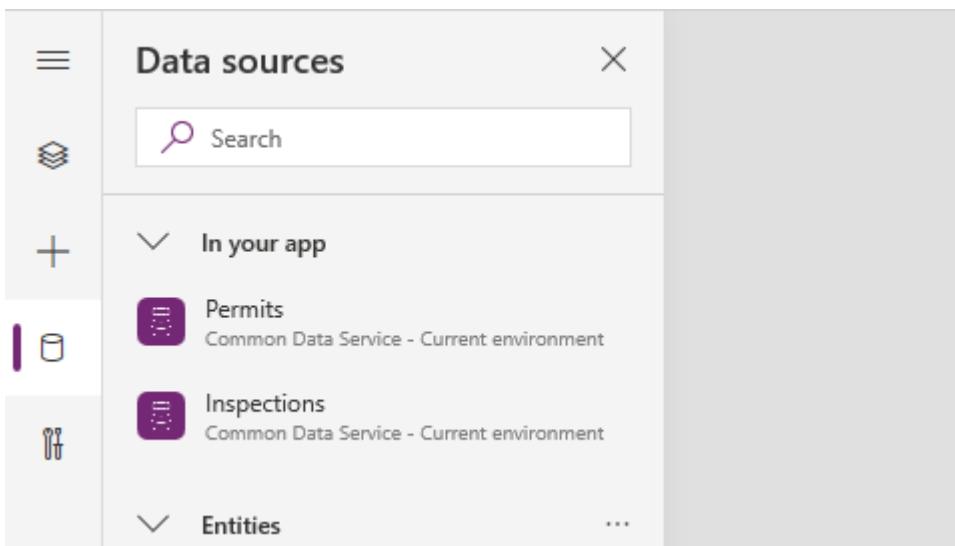
- Search for **Permit** and select **Permits**.

The screenshot shows the "Data sources" screen. At the top, there is a search bar with "Permit" entered. Below it, two items are listed: "Permit Types" and "Permits". A red arrow points to the "Permits" item, which is described as "Common Data Service".

- Search for **Inspection** and select **Inspections**.



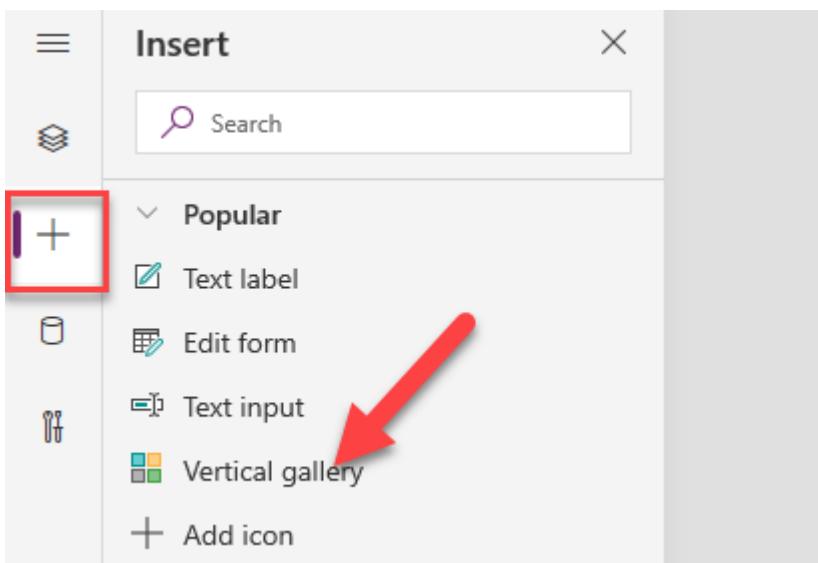
- You should now have both **Permits** and **Inspections** in your app.



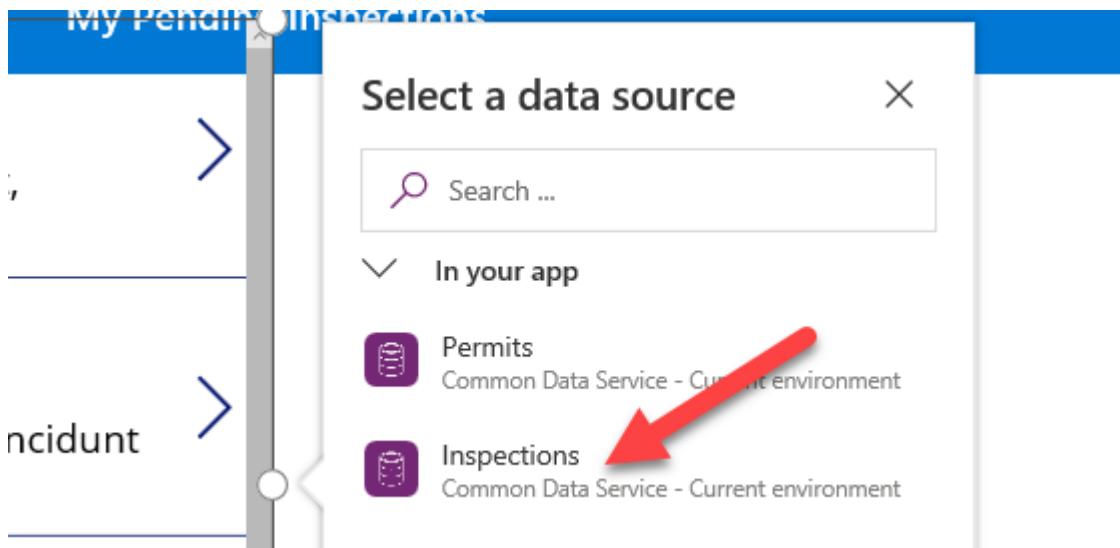
15.3 Task #3: Add Inspection Gallery

1. Add Gallery

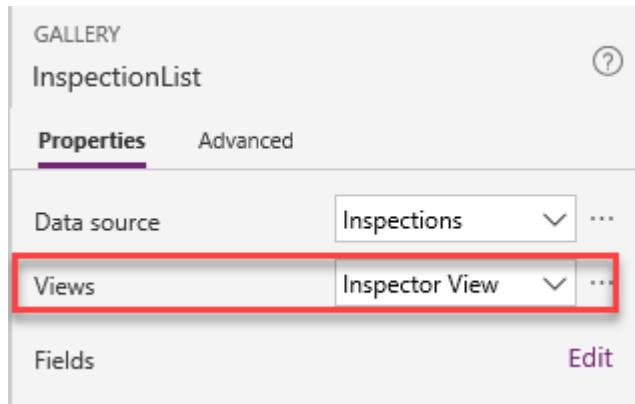
- Click **+ Insert** and select **Vertical Gallery**. Vertical Gallery will be added to the MainScreen.



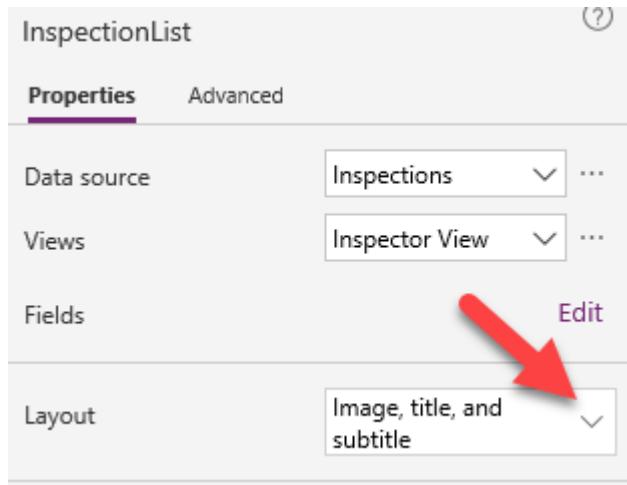
- Select **Inspections** for **Data Source**. When Inspection is selected, this will automatically pick the Columns and show them in the gallery items.



- Select the **Tree View** tab.
 - Rename **Gallery_1** to **Inspection List** by double click on **Gallery_1**.
2. Select the inspector view
- Make sure you have the **Inspection List** control selected.
 - Go to the **Properties** pane and select **Inspector View** for **View**.

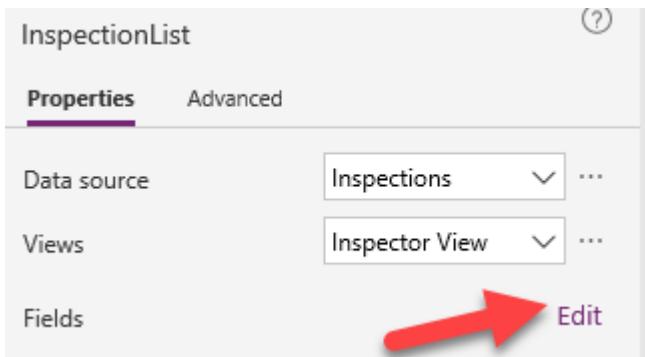


3. Change the **Inspection List** control layout
- Go to the **Properties** pane and click on the **Layout** dropdown.



- Select **Title and Subtitle**.
4. Verify the selected Columns.

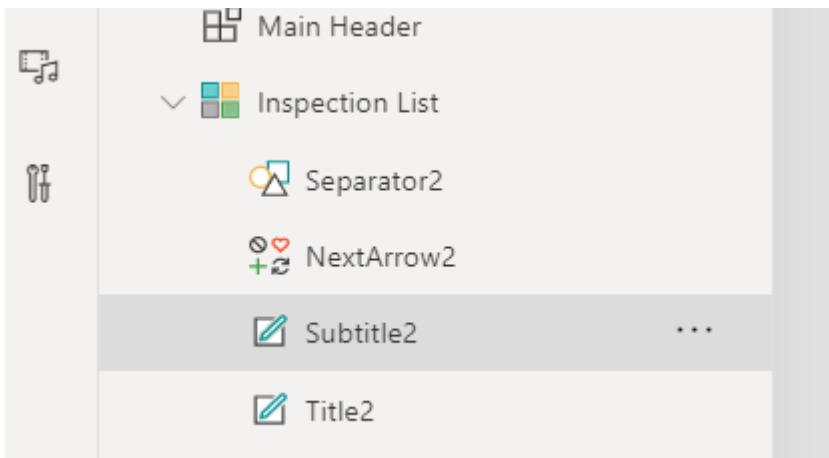
- Go to the **Properties** pane and click on the **Edit Columns**.



- Confirm that **Scheduled Date** is selected for **Subtitle** and **Name** is selected for **Title**. Close the **Data** pane.

5. Change date time to date only.

- Expand the **Inspection List** and select **Subtitle**.



- Change the **Text** property of the control to the formula below.

```
DateValue(Text(ThisItem.'Scheduled Date'), "en")
```

6. Resize the Gallery

- Select the **Inspection List** gallery.
- Select **Width** property from the formula dropdown and enter the formula below.

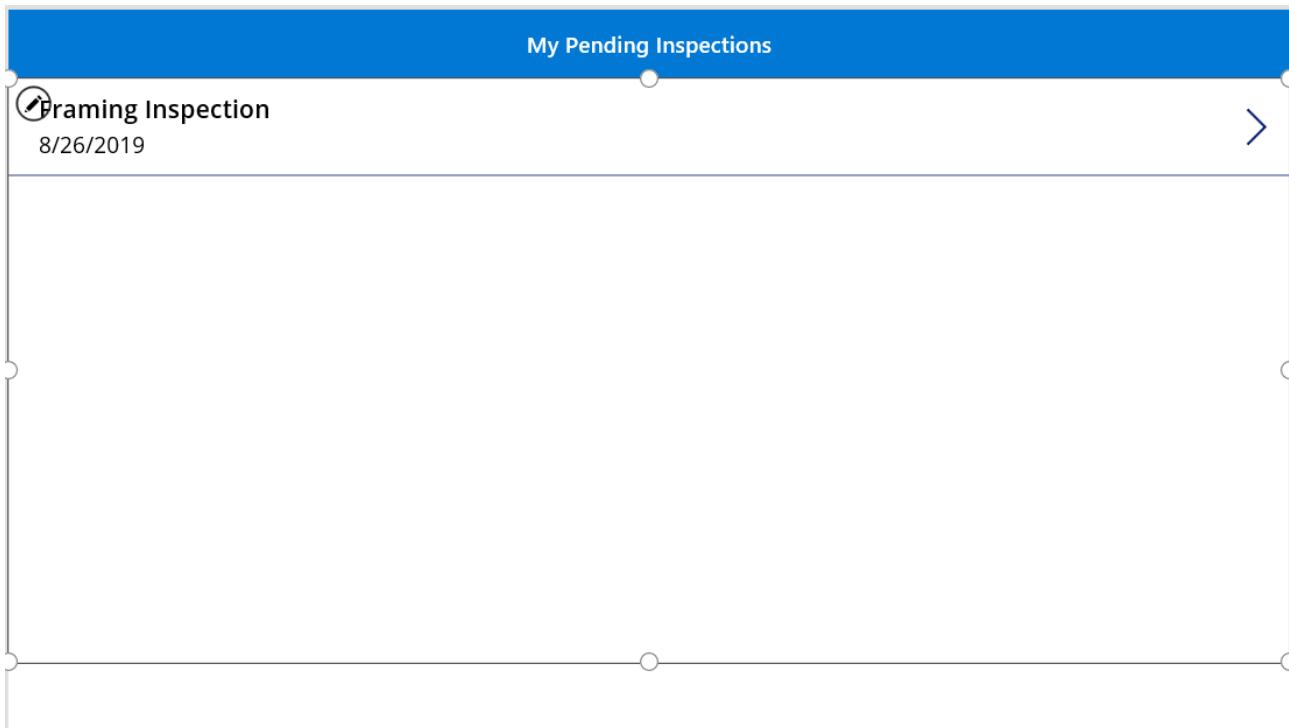
Parent.Width
- Select **Height** property and set it to the formula below.

Parent.Height - ('Main Header'.Height*2)
- Select the **Y** property from the dropdown and set it to formula below.

'Main Header'.Height

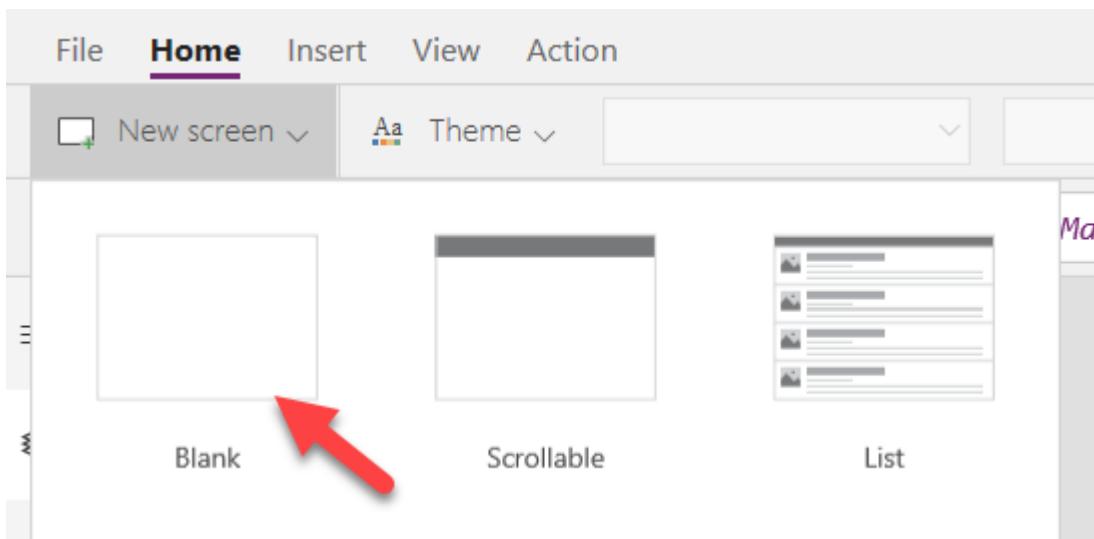
- Select the X property from the dropdown and set it to formula below.

'Main Header'.X

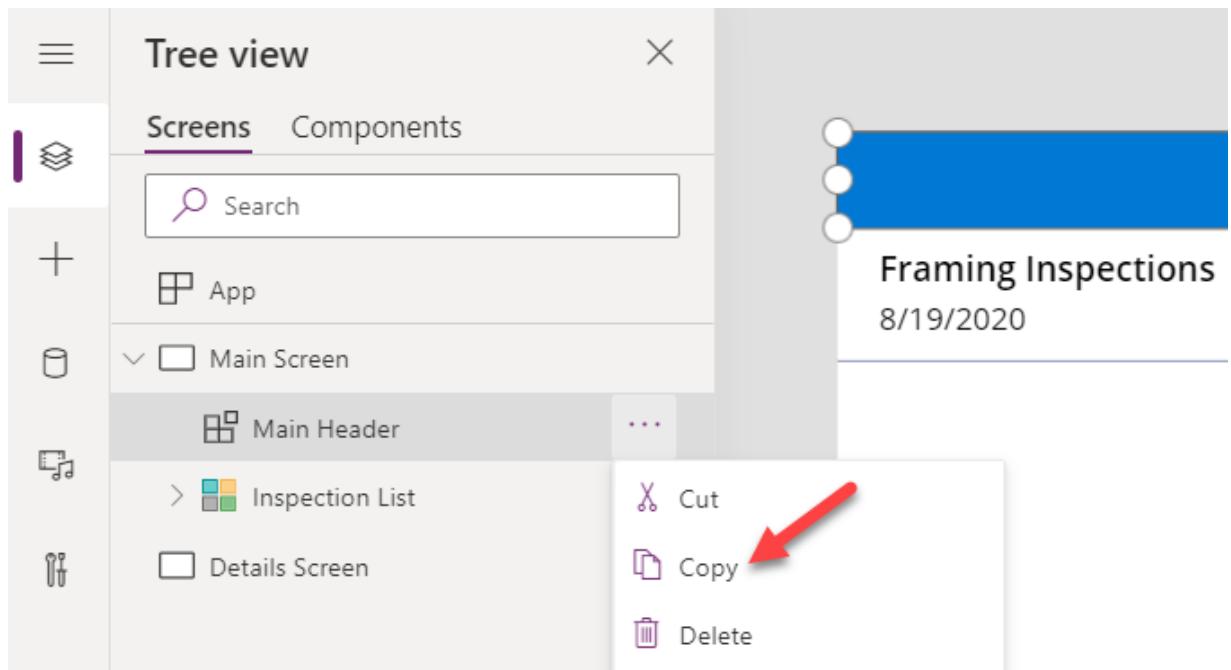


15.4 Task #4: Add Inspection Details Screen

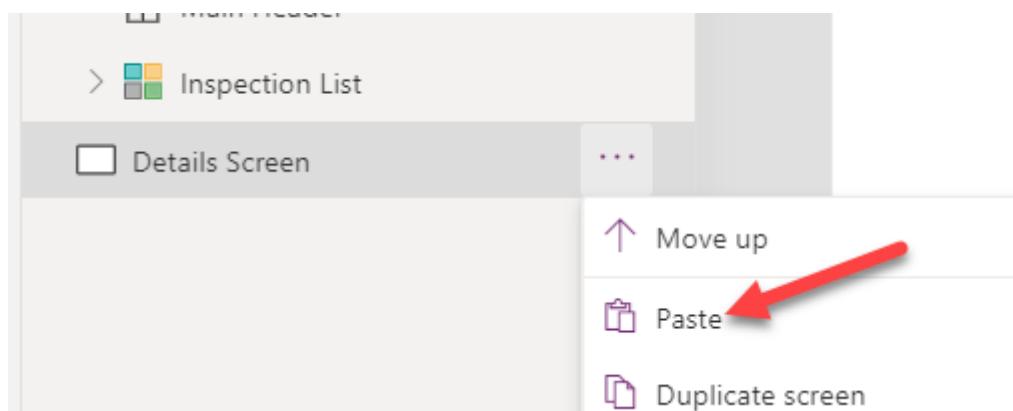
1. Add new screen named Details Screen
 - Click **New Screen** and select **Blank**.



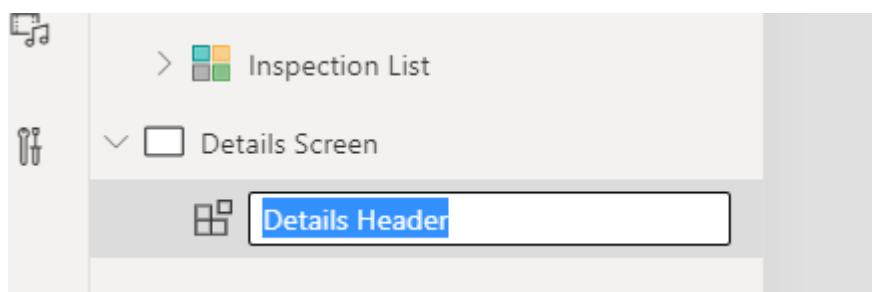
- Rename the new screen **Details Screen** by double clicking on the control in Tree View.
2. Add Header to the Details Screen and edit it
 - Go to the **Main Screen** and copy the **Main Header**.



- Go to the **Details Screen** and paste the **Header**.

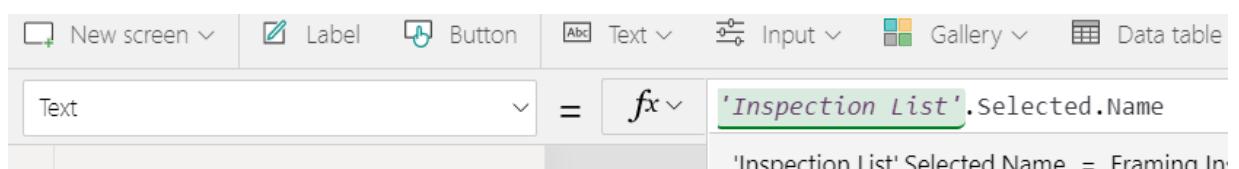


- Rename the Header you **Details Header** by double clicking on the control in Tree View.



- Select te **Y** property of the **Details Header** and set to **0**.
- Select the **Text** property of the **Details Header** and set it to formula below.

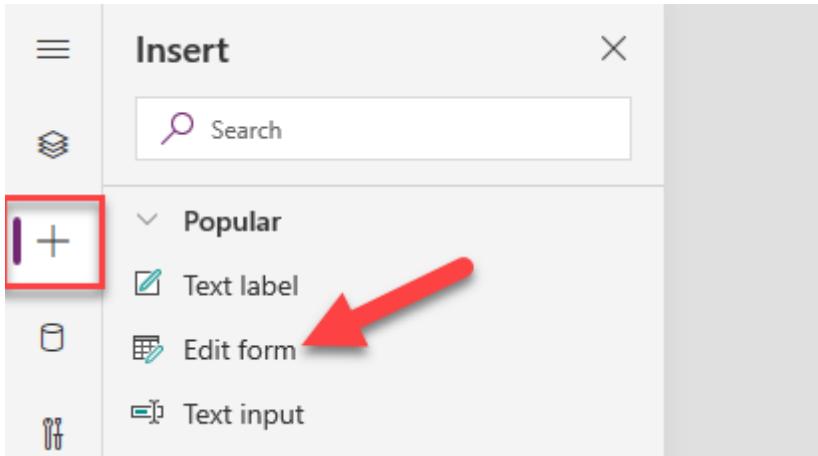
`'Inspection List'.Selected.Name`



3. Add Form to the Details Screen.

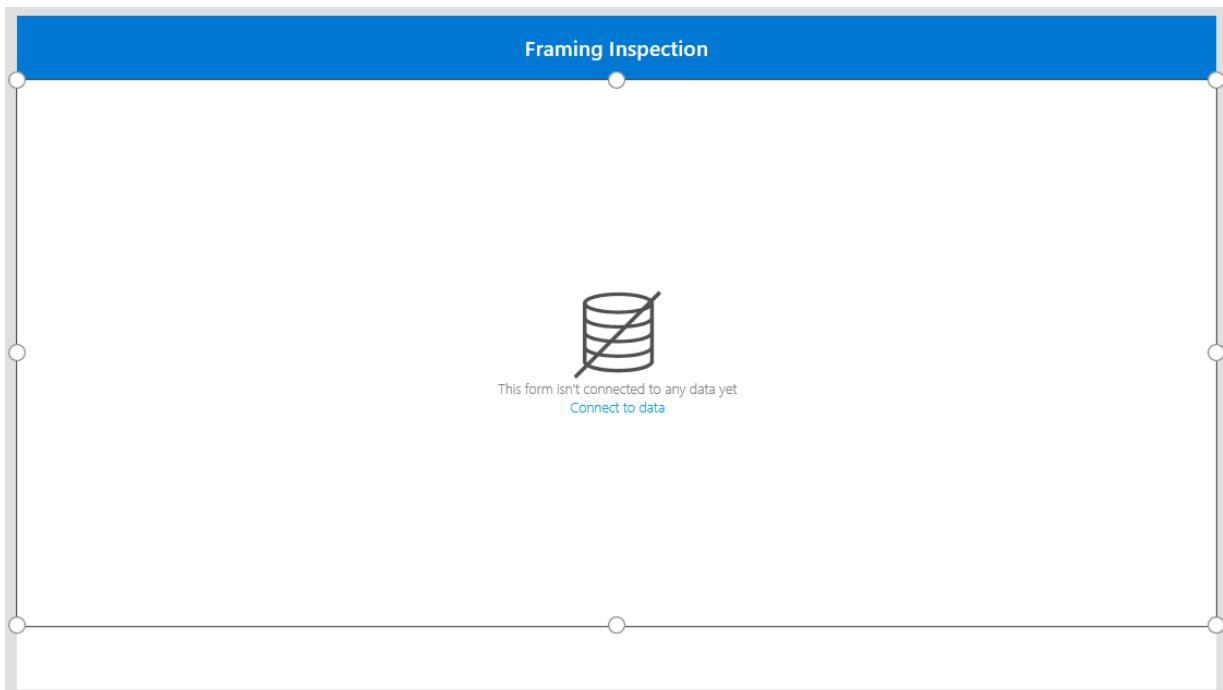
- Select the **Details Screen**.

- Click **+ Insert** and select **Edit Form**.



- Select the **Tree View** tab.
- Rename the form **Inspection Form**.
- Resize the Edit form as:
 - Select the **InspectionForm**.
 - Select **Width** property from the formula dropdown and enter the formula below.

```
Parent.Width
```
- Select **Height** property and set it to the formula below.
- Select the **Y** property from the dropdown and set it to formula below.
- Select the **X** property from the dropdown and set it to formula below.
- The form should now look like the image below.



4. Set the **Inspection Form** data source

- Select the **Inspection Form** and select the Data Source as **Inspections Table**.

The screenshot shows the Power BI formula bar. The formula is: `DataSource = fx Inspections`. To the right, it says "Inspections Data type: Table". Below the formula bar, there's a "Tree view" pane with tabs for "Screens" and "Components".

- Set the Item value to the formula below.

```
'Inspection List'.Selected
```

The screenshot shows the Power BI formula bar. The formula is: `Item = fx InspectionList.Selected`. To the right, it says "InspectionList.Selected". Below the formula bar, there's a "Tree view" pane with tabs for "Screens" and "Components".

- Edit Inspection Form Columns. This adds the data cards for Columns by default, but you can add/remove the data cards as:

- Select the **Inspection Form**.
- Go to the **Properties** pane and click **Edit Columns**.

The screenshot shows the Power BI Properties pane for "InspectionForm". The "Properties" tab is selected. Under "Data source", it shows "Inspections". In the "Fields" section, there is a red arrow pointing to the "Edit fields" link.

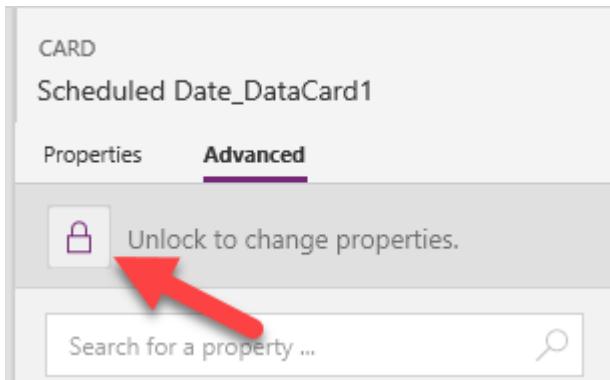
- Click **Add Column**.
- Select **Status Reason**, and **Comments**.
- Click **Add**.
- The Columns should be ordered as shown in the image below. You can drag/drop to rearrange the Columns.

The screenshot shows the Power BI Fields pane. It lists four columns: "Name", "Scheduled Date", "Status Reason", and "Comments". A red arrow points to the "X" icon at the top right of the Fields pane, indicating how to close it.

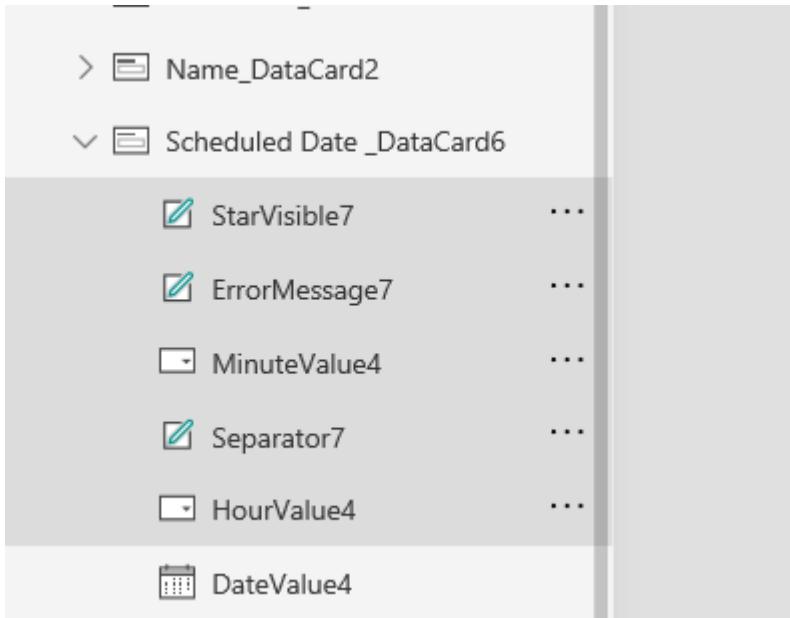
- Change the Scheduled Date to show date only.

- Go to the **Tree View** and expand the **Inspection Form**.

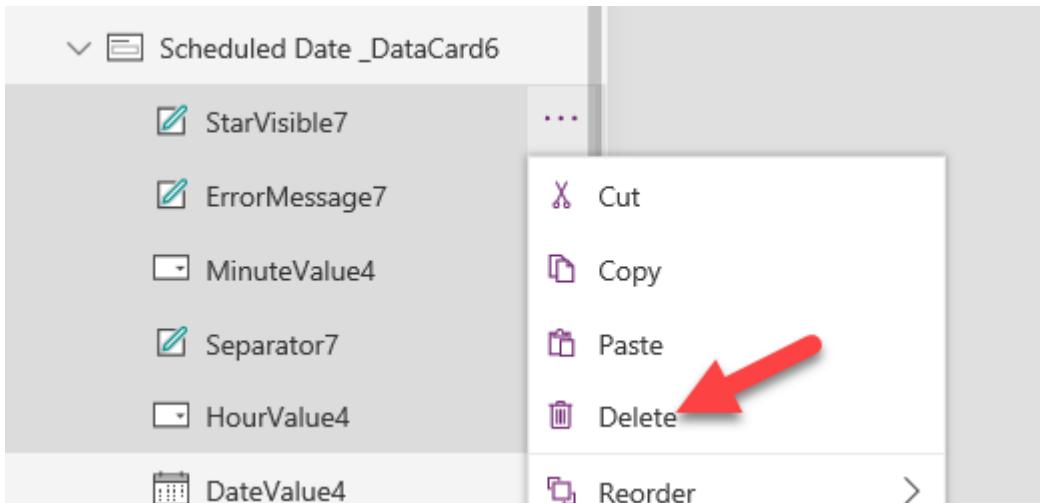
- Select the **Scheduled Date** data card.
- Go to the **Properties** pane and select the **Advanced** tab.
- Click **Unlock to change Properties**.



- Expand the **Scheduled Date** card.
- Select **StarVisible**, **ErrorMesage**, **MinuteValue**, **Separator**, and **HourValue**.



- Delete the selected controls. When the controls are deleted, you will see an error message.



- Select the **Scheduled Date** DataCard.
- Go to formula bar and select **Update**.

The screenshot shows the Power Apps formula bar with the formula `DateValue4.SelectedDate + Time(Value(HourValue4.Selected.Value), Value(MinuteValue4.Selected.Value), 0)`. The entire formula is highlighted with a red box.

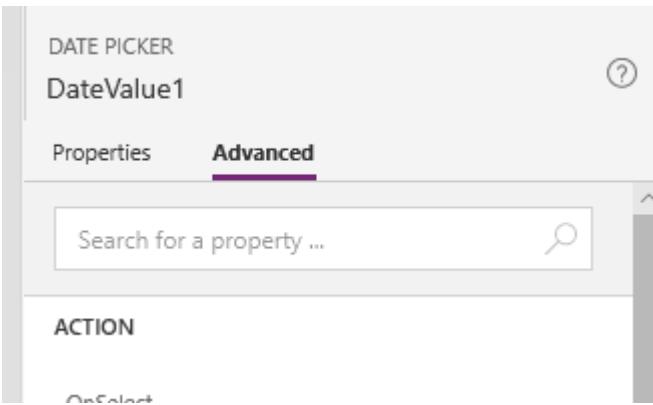
- Remove everything after the **SelectedDate**. This should remove the error message from the app.

- Make the Name and Scheduled Date Columns read-only
 - Select the **Inspection Form**.
 - Go to the **Properties** pane and click **Edit Columns**.
 - Expand the **Name** Column.

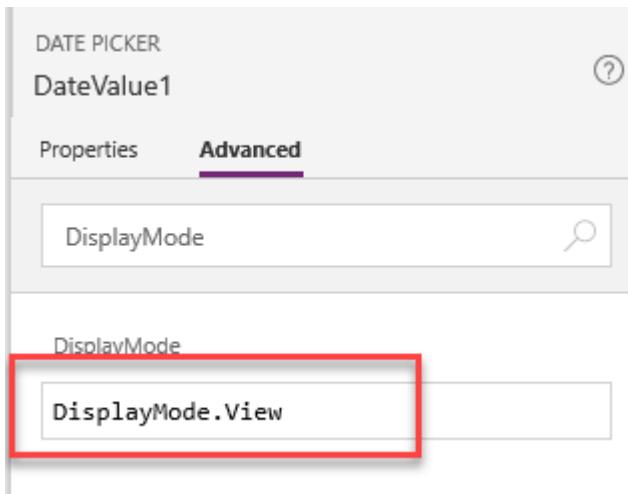
Fields

The screenshot shows the Fields pane with the **Name** column selected. A red arrow points to the **Control Type** dropdown menu, which is open to show options: **View text**, **View phone**, and **View email**.

- Expand the **Scheduled Date** Column. Observe the change.
- Notice we cannot change this the same way because we've customized it. From the Tree View select **DateValue** control inside the **Scheduled Date** Datacard and go to the **Advanced tab** of the **Properties** pane.

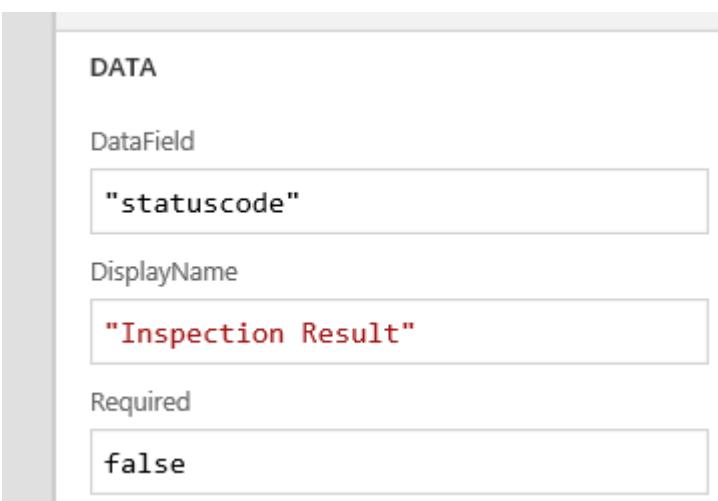


- Search for **DisplayMode** property and remove the existing formula and replace it with the following:
 DisplayMode.View



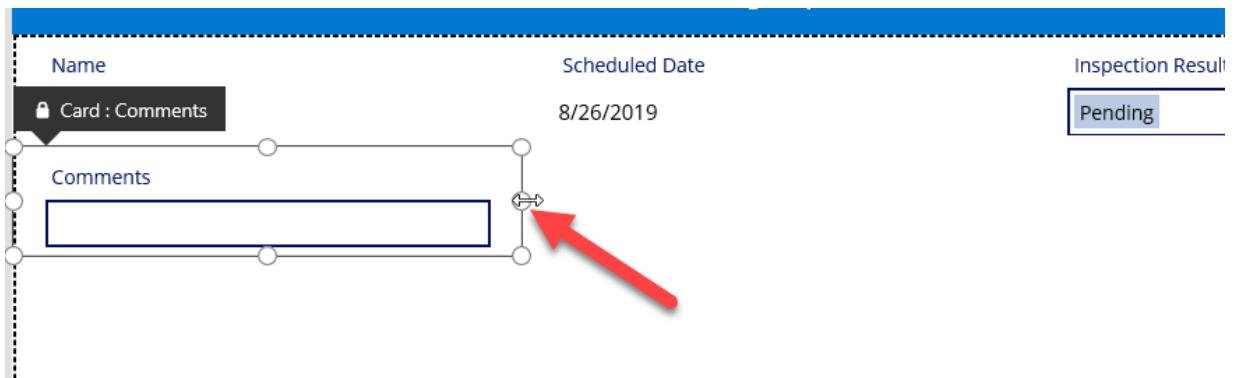
8. Change the Status Reason label.

- Select the **Status Reason** data card.
- Go to the **Properties** pane and the Advanced tab, click **Unlock to Change properties**.
- Change the **DisplayName** to **Inspection Result**.



9. Resize the Comments data card.

- Select the **Comments** data card.
- Click and drag the right edge to the far right of the screen.



- Go to the **Advanced** tab of **Properties** pane and click **Unlock** to change properties.
- Set the **Height** value to **300**.
- Select the **DataCardValue** control.

The screenshot shows the PowerApps Properties pane with the following structure:

- Status Reason_DataCard1** (expanded)
- Comments_DataCard1** (expanded)
 - StarVisible4**
 - ErrorMessage4**
 - DataCardValue3** (selected, highlighted with a red arrow)
 - DataCardKey4**
- Scheduled Date DataCard6** (collapsed)

- Set the **Height** value to **300**.
- Change the **Mode** to the formula below.

The screenshot shows the PowerApps formula bar with the following formula:

```
TextMode.MultiLine
```

The formula bar includes the following elements from left to right:

- New screen
- Label
- Button
- Text
- Controls
- Gallery
- Mode dropdown: TextMode.MultiLine
- Mode dropdown: TextMode.MultiLine = m
- Tree view button
- X button

10. Make sure your form looks like the image below. Save your work.

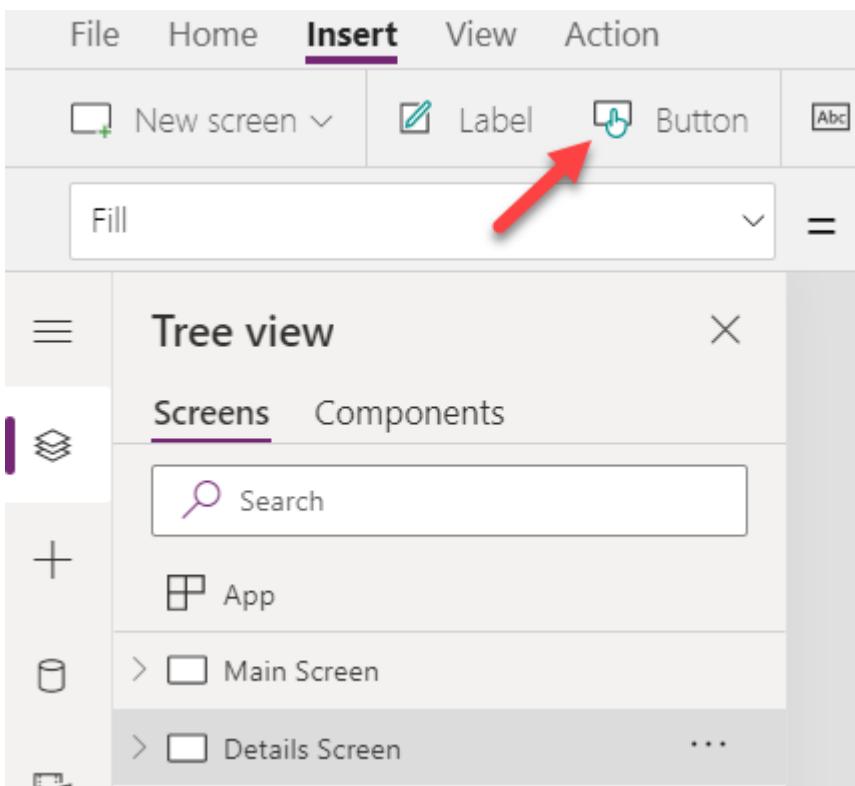
Framing Inspection

Name	Scheduled Date	Inspection Result
Framing Inspection	8/26/2019	Pending
Comments		

15.5 Task #5: Submit the Inspection Result

1. Add submit button to the details screen.

- Select the **Details Screen**. Make sure that you have selected the screen and not selected the Edit Form.
- Go to the **Insert** tab and click **Button**.



- Rename the button **Submit Button**.
- Change the Text value of the button to **Submit**.
- Place the button below the form through drag and drop.

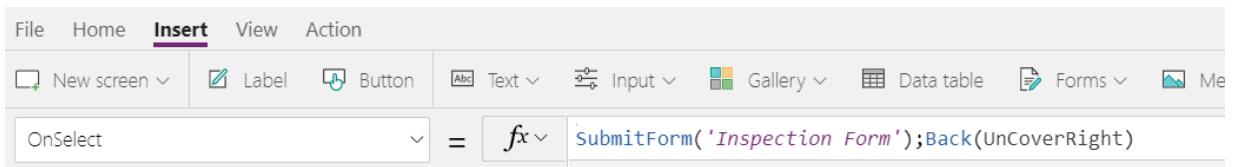
Framing Inspection

Name	Scheduled Date	Inspection Result
Framing Inspection	8/26/2019	Pending
Comments		
<input type="button" value="Submit"/>		

2. Submit the inspection result.

- Select the **Submit Button**.
- Set the **OnSelect** value of the submit button to the formula below. Remove the false expression and update it. This formula will submit the form and then navigate back to the MainScreen.

```
SubmitForm('Inspection Form');Back(ScreenTransition.UnCoverRight)
```



3. Add navigation from the main screen to the details screen.

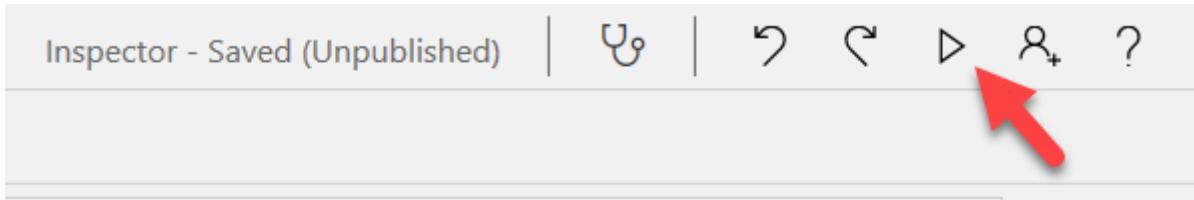
- Go to the **Main Screen** and select the **Inspection List**.
- Set the **OnSelect** property of the **Inspection List** to the formula below. Remove the already existing false expression.

```
Navigate('Details Screen', ScreenTransition.Cover)
```

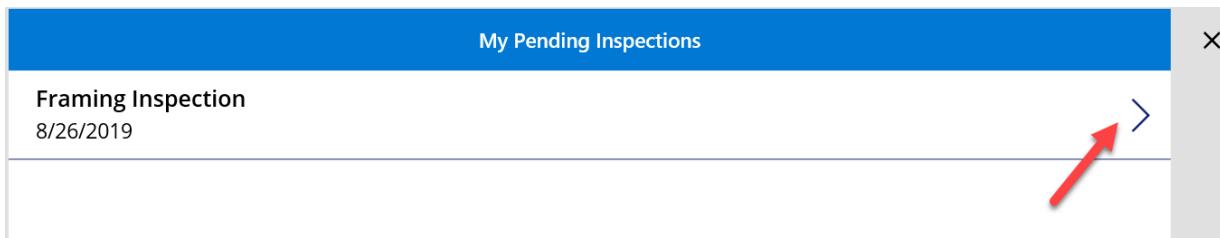
15.6 Task #6: Test Application

1. Start the application

- Select the **Main Screen** and click **Preview the App**.



- The application should load and show at least one inspection. Click on the inspection.



- The application should navigate to the details screen. Change the **Inspection Result** to **Passed**, provide a comment in the textbox as “Framing inspection was completed.”, and click **Submit**.

Framing Inspection

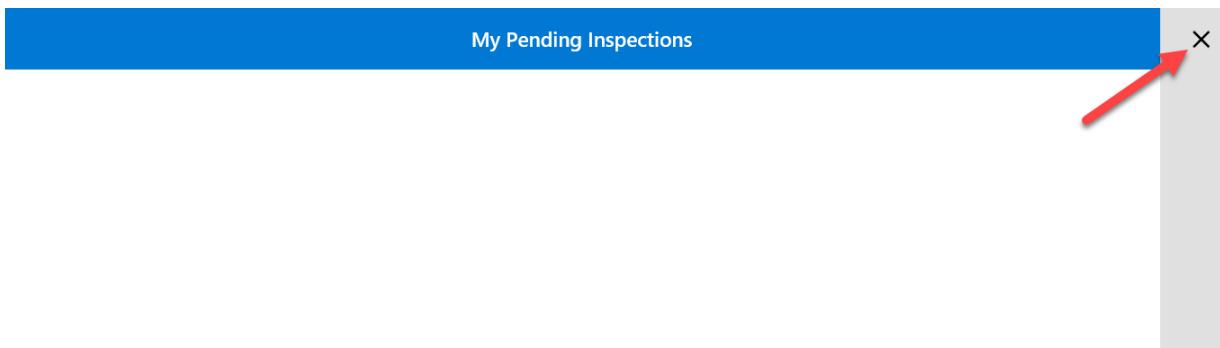
Name	Scheduled Date	Inspection Result
Framing Inspection	8/26/2019	Passed

Comments

Framing inspection was completed.

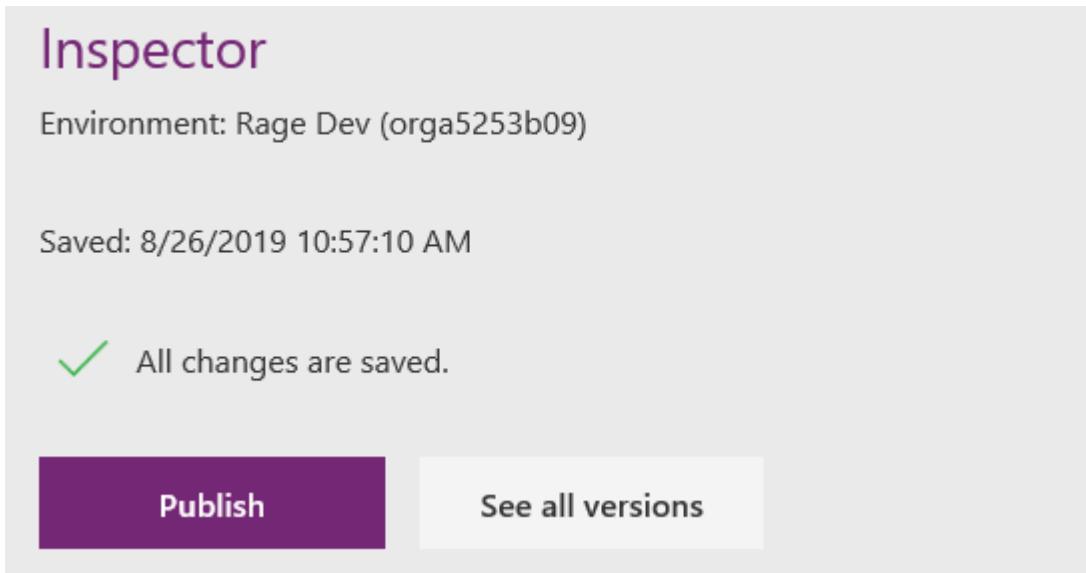
Submit

- The inspection should be submitted, and the application should navigate back to the MainScreen. Click Close.

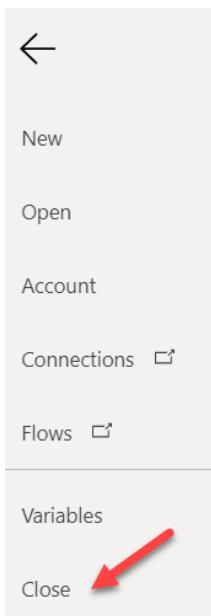


2. Save and publish the application

- Click **File** and then click **Save**.
- Click **Publish**.



- Click **Publish this Version**.
- Click **Close**.



Inspector

Environment: Rage Dev (ragedev)

Saved: 8/12/2020, 10:23:59 AM

✓ All changes are saved and published.

Share

See all versions

- Close the **Designer** browser window or tab.
 - Click **Leave** if prompted when trying to close the browser window.
3. Confirm the inspection record was updated
- Navigate to **Power Apps maker portal**
 - Select your **Dev environment**.
 - Select **Apps** and click to open the **Permit Management Application**.

A screenshot of the Power Apps maker portal. On the left is a sidebar with Home, Learn, Apps (selected), Create, Data, Flows, Chatbots, and a gear icon. The main area shows the title 'Apps' and a list of applications: Inspector, Permit Management (highlighted with a red arrow), and Solution Health Hub.

- Select **Inspections** and click to open the **Framing Inspection**.

The screenshot shows the 'Active Inspections' list in the Power Apps maker portal. The left sidebar has 'Permits' selected. The main area shows a table with one row for 'Framing Inspection'. The 'Name' column contains 'Framing Inspection' with a red arrow pointing to it. The 'Inspection Type' column shows 'Initial Inspection'. The right side of the table includes 'Status Reason' and 'Passed'.

- The **Status Reason** of the inspection should be **Passed**, and the comment should be updated to the comment you provided.

The screenshot shows the details for the 'Framing Inspection' record. It includes a circular icon with a document symbol, the title 'INSPECTION', the name 'Framing Inspection', the status 'Status Reason Passed', and tabs for 'General' and 'Related'.

Name	Framing Inspection
Inspection Type	Initial Inspection
Permit	Test Permit
Scheduled Date	8/26/2019
Owner	MOD Administrator
Comments	Framing inspection was completed.

- Close the **Permit Management** application.

16 Exercise #2: Export/Import Solution

Objective: In this exercise, you will export the solution you created in the development environment and import it to the production environment.

16.1 Task #1: Export solution.

1. Select the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)
 - Make sure you have your **Dev** environment selected.
 - Select **Solutions** and select the **Permit Management** solution.

The screenshot shows the 'PowerApps Checker Update' interface. On the left, there's a sidebar with options: 'Flows', 'AI Builder (preview)', and 'Solutions'. Below these are three solution cards: 'Permit Management' (selected, highlighted in grey), 'Common Data Services Default Solution', and another partially visible solution.

2. Run solution checker.

- Click **Solution Checker** and select **Run**.

The screenshot shows the 'Solution checker' interface. At the top, there are buttons for 'Export', 'Solution checker' (which is selected and highlighted in grey), 'Show dependencies', and more. Below this is a large 'Run' button, which has a red arrow pointing to it.

- Wait for the run to complete.

Note: At the creation of this lab, the solution checker did not complete successfully. If you get an error with the message “Couldn’t be complete” skip to step 3.

The screenshot shows the 'Solution checker running' status at the top. Below it, a message says 'This may take a few minutes. We'll notify you when it's ready to download.' A detailed view of the 'Permit Management' solution card is shown, including its creation date (8/22/2019) and version (1.0.0.2). A red arrow points to the 'More Commands' button (three dots) on the right side of the card.

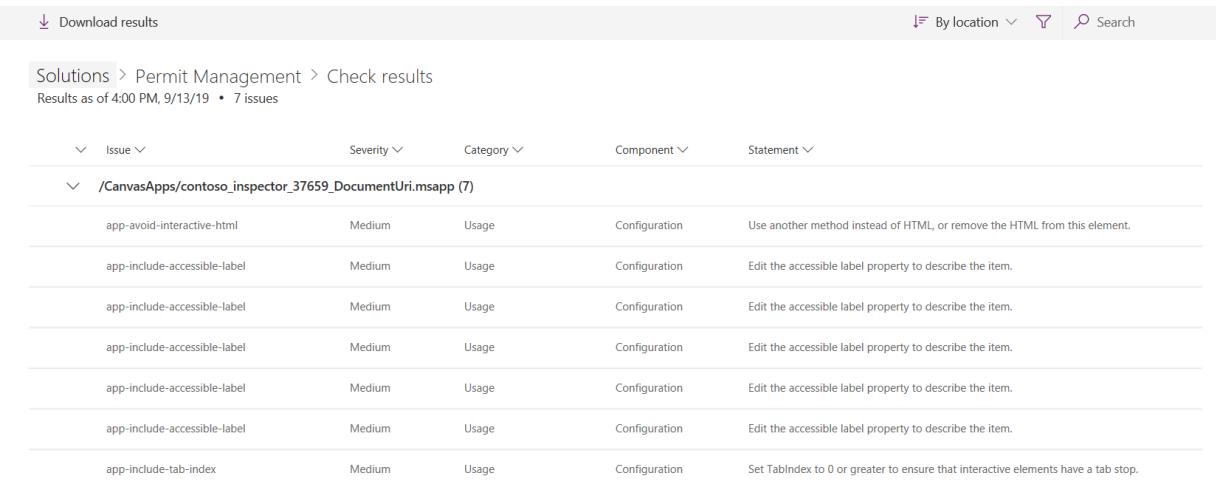
- Click on the More **Commands** of the **Permit Management** solution.

The screenshot shows the 'More Commands' menu for the 'Permit Management' solution. The menu items are: Edit, Delete, Export, Solution checker (which is selected and highlighted in grey), Show dependencies, Clone, Run, View results, and Download results. A red arrow points to the 'View results' option.

- Click Solution Checker and select View Results.

The screenshot shows the 'More Commands' menu for the 'Permit Management' solution again. The 'View results' option is highlighted with a red arrow. The other menu items are: Edit, Delete, Export, Solution checker (selected), Show dependencies, Clone, Run, View results (highlighted), and Download results.

- You will see several issues reported



Solutions > Permit Management > Check results
Results as of 4:00 PM, 9/13/19 • 7 issues

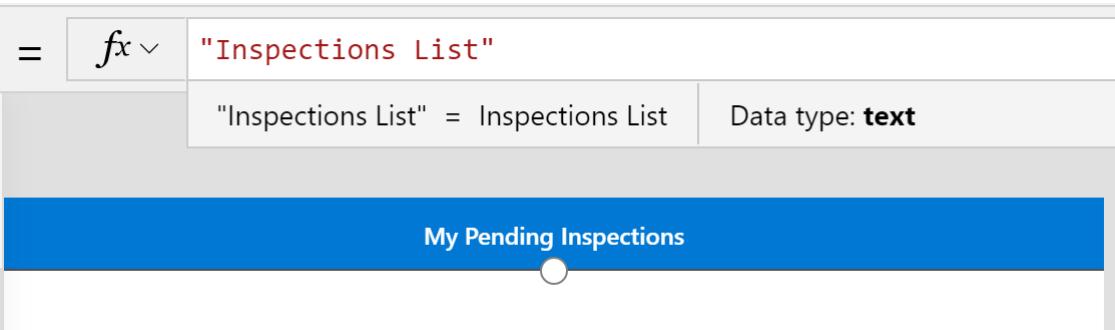
Issue ▾	Severity ▾	Category ▾	Component ▾	Statement ▾
▽ /CanvasApps/contoso_inspector_37659_DocumentUri.msapp (7)				
app-avoid-interactive-html	Medium	Usage	Configuration	Use another method instead of HTML, or remove the HTML from this element.
app-include-accessible-label	Medium	Usage	Configuration	Edit the accessible label property to describe the item.
app-include-accessible-label	Medium	Usage	Configuration	Edit the accessible label property to describe the item.
app-include-accessible-label	Medium	Usage	Configuration	Edit the accessible label property to describe the item.
app-include-accessible-label	Medium	Usage	Configuration	Edit the accessible label property to describe the item.
app-include-accessible-label	Medium	Usage	Configuration	Edit the accessible label property to describe the item.
app-include-tab-index	Medium	Usage	Configuration	Set TabIndex to 0 or greater to ensure that interactive elements have a tab stop.

- To resolve the issues, follow these steps:

- Select **Apps**
- Click ... next to **Inspector** app and select **Edit**
- Click **App checked** icon on the toolbar



- Select **Recheck All**
- Expand **Missing accessible label** node
- Select an issue. This will open the screen with the control and prompt to enter **AccessibleLabel** property.
- Enter text value as appropriate



"Inspections List" = Inspections List Data type: **text**

My Pending Inspections

- Repeat the process for all controls with missing accessible labels
- Expand **Missing tab stop** node
- Select control, enter a value for the **TabIndex**, e.g. 0
- **Tips** node may contain the following message
“Use another method instead of HTML, or remove the HTML from this element.”
- This message is related to Map component we imported as part of the component bundle. This component can be safely deleted as it's not used by the app.
- Fix other app issues as appropriate.
- Click **File** and then click **Save**.
- Click **Publish**.

- Switch to Power Apps maker portal
- Select **Solutions** then select **Permit Management** solution
- Click **Solution checker** then select **Run** and wait for the run to complete.
- There should be zero issues.

Solutions > Permit Management > Check results

Results as of 1:28 PM, 8/26/19 • 0 issues

Issue ▾

Severity ▾

Category ▾

3. Export managed solution

- Select **Solutions** and click to open the **Permit Management** solution.
- Click **Export**.



Solutions > Permit Management

Display name	Name	Type	Manage
Contact	... contact	Entity	🔒

- Click **Publish** and wait for the publishing to complete.

Before you export



Publish all changes

If you made changes to this solution that you'd like to export, publish them now.
[Learn more](#)

Publish

Check for issues

0 found on 08/26/2019

- Click **Next**.
- Select **Managed** and click **Export**.

Version number* ⓘ

Current version 1.0.0.0

1.0.0.1

Export as

Managed (recommended) ⓘ

The solution is moving to a test or production environment. [Learn more](#)

Unmanaged

The solution is moving to another development environment or source control. [Learn more](#)

Export

Cancel

- Save the solution on your machine.

4. Export unmanaged solution

- Click **Export** again.
- Click **Next**.
- Edit the version number to match the Managed solution you just exported, select **Unmanaged** and click **Export**.

← [Export this solution](#) ×

Version number* ⓘ

Current version 1.0.0.1

1.0.0.1

Export as

Managed (recommended) ⓘ

The solution is moving to a test or production environment. [Learn more](#)

Unmanaged

The solution is moving to another development environment or source control. [Learn more](#)

- Save the unmanaged solution on your machine.

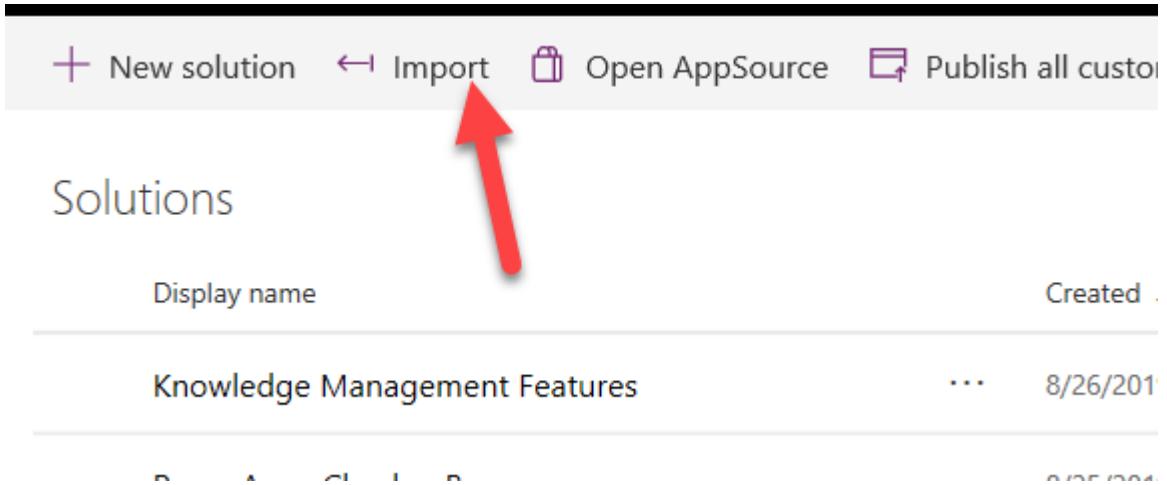
16.2 Task #2: Import solution.

1. Open the Permit Management solution.
 - Sign in to [Power Apps maker portal](#)

- Make sure you have your **Prod** environment selected. **Note** If you are using the community plan environment as instructed (e.g: [username]'s Environment), you may need to create a data base before you proceed to the next step.

2. Import solution

- Select **Solutions** and click **Import**.



The screenshot shows the 'Solutions' blade in the Microsoft Power Platform. At the top, there are several buttons: 'New solution', 'Import' (which has a red arrow pointing to it), 'Open AppSource', and 'Publish all custom...'. Below the buttons, the word 'Solutions' is displayed. A table lists a single solution entry:

Display name	Created
Knowledge Management Features	8/26/2019

Below the table, there are navigation icons for back, forward, and search. Further down, there is a file list and a file selection dialog box:

File list:

PermitManagement_1_0_0_1_managed.zip	8/26/2019 11:17 AM	Compressed (zipp...)	101 KB
PermitManagement_1_0_0_2.zip	8/26/2019 11:19 AM	Compressed (zipp...)	110 KB

File selection dialog:

File: PermitManagement_1_0_0_1_managed.zip

Filter: All files (*)

Buttons: Open, Cancel

- Click **Browse**.
- Select the **Managed** solution you exported and click **Open**.
- Click **Next**.
- Click **Import** and this will open a new window to track the import status.
- Wait for import to complete and click **Close**.
- Navigate to both the model driven and canvas apps you've created and add a few records, test the apps.

16.3 lab: title: 'Lab 04: Client Scripting'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

16.4 Lab 04 – Client Scripting

17 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course, you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab, you will implement client-side logic that will use the web API to evaluate the permit type associated with the permit record and use the client scripting API to manipulate the form controls.

You will also customize the command bar to introduce a new lock permit button that will invoke a custom action to perform the lock permit logic. The server-side logic for the lock permit custom action will be implemented later in the course. Right now, you will just add the button and the logic to invoke the action.

17.1 High-level lab steps

As part of building the client-side logic, you will complete the following:

- Setup a folder to contain your client script
- Upload and register the client script on the form
- Build logic to use the web API to retrieve the permit type record associated with the permit
- Build logic based on the permit type settings to hide and show the inspections tab on the form
- Build logic to set Columns as required/not required based on the permit type settings
- Use a community tool, Ribbon Workbench, to modify the command bar
- Build logic to invoke the lock permit custom action when the command bar button is clicked

17.2 Things to consider before you begin

- Are there alternative designs that would be viable and not require code?
- Remember to continue working in your DEVELOPMENT environment. We will move everything to production soon.

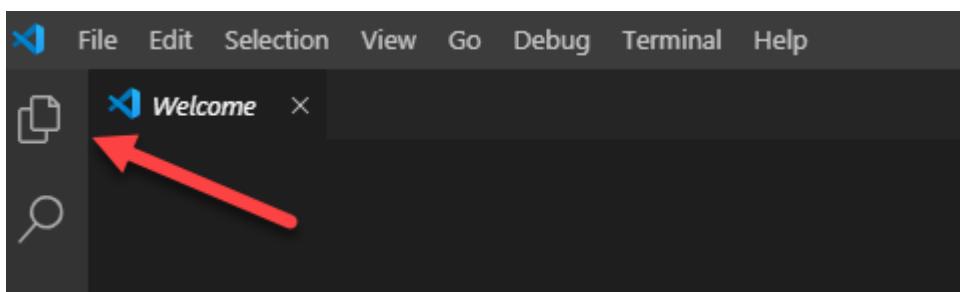
18 Exercise #1: Prepare and Load Resources

Objective: In this exercise, you will create, organize, and load your JavaScript web resources.

18.1 Task #1: Use Visual Studio Code to Create Resources

In this task, you will set up a folder to contain the JavaScript web resource files in this course.

1. If you do not already have Visual Studio Code, download it from here [Visual Studio Code](#) and install it.
2. Start **Visual Studio Code**.
3. Create resources
 - Select Explorer from left menu or press Ctrl + Shift + E.

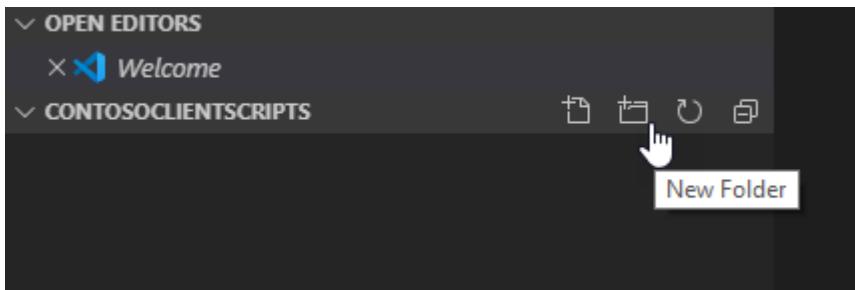


- Click **Open Folder**.
- Create a new folder and name it **ContosoClientScripts**.
Note: This is the name and structure used for this lab, the platform does not require a specific structure or content organization. Many projects check these assets into a source control system to keep track of all the changes over the life of the client script.
- Select the new folder you just created and click **Select Folder**.

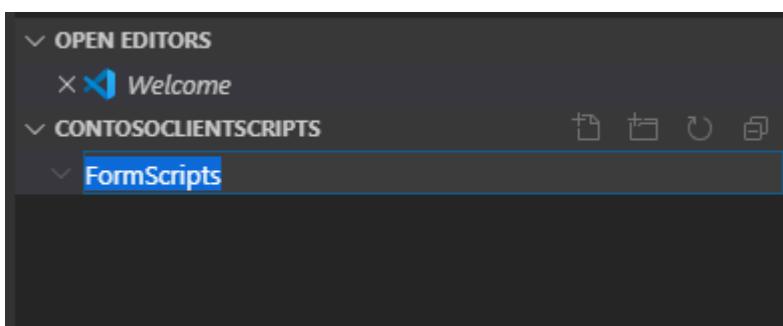


4. Create **Form Scripts** folder

- Hover over the folder and click **New Folder**.

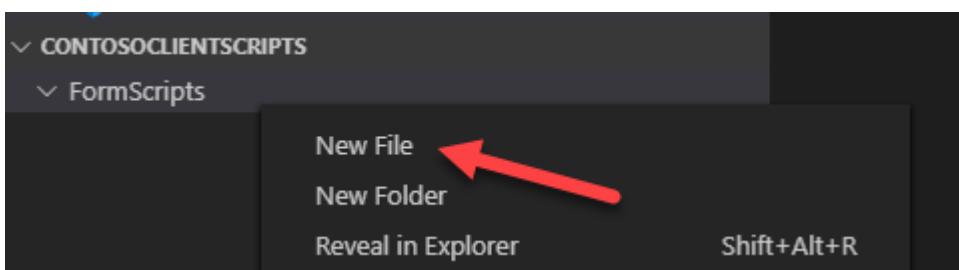


- Name the new folder as **FormScripts** and click **Enter**.

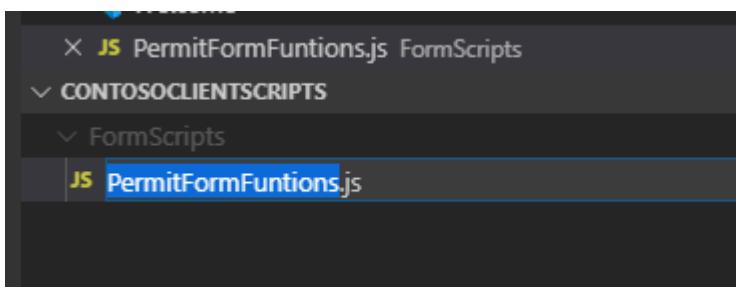


5. Create the **Permit Form Functions** file

- Right click on the **FormScripts** folder and select **New File**.



- Name the new file as **PermitFormFuntions.js** and click **Enter**.



- Add the below mentioned namespaces to the newly created **PermitFormFunctions** file.

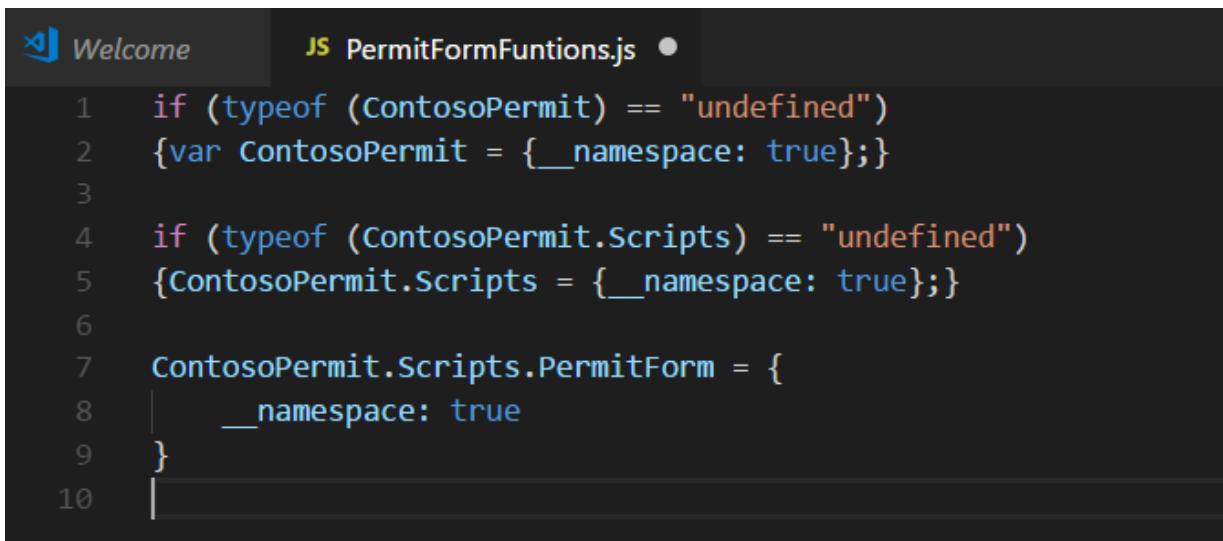
```
if (typeof (ContosoPermit) == "undefined")
{var ContosoPermit = {__namespace: true};}
```

```
if (typeof (ContosoPermit.Scripts) == "undefined")
{ContosoPermit.Scripts = {__namespace: true};}
```

```
FormScripts > JS PermitFormFuntions.js > ...
1  if (typeof (ContosoPermit) == "undefined")
2  {var ContosoPermit = {__namespace: true};}
3
4  if (typeof (ContosoPermit.Scripts) == "undefined")
5  {ContosoPermit.Scripts = {__namespace: true};}
6
7
```

- Add the function mentioned below after adding the namespaces.

```
ContosoPermit.Scripts.PermitForm = {
  __namespace: true
}
```



```
1  if (typeof (ContosoPermit) == "undefined")
2  {var ContosoPermit = {__namespace: true};}
3
4  if (typeof (ContosoPermit.Scripts) == "undefined")
5  {ContosoPermit.Scripts = {__namespace: true};}
6
7  ContosoPermit.Scripts.PermitForm = {
8  |   __namespace: true
9  }
10 |
```

18.2 Task #2: Add Event Handlers

In this task, you will create functions for the logic that you will be implementing. This will allow you to register the event handlers in the next tasks for calling these functions and performing few basic tests in the upcoming tasks.

1. Add a function to OnLoad event

- Add the function mentioned below to the **PermitFormFuntions** file inside the function created in Step 1(d).

```
handleOnLoad: function (executionContext) {
  console.log('on load - permit form');
},
```

```

6
7 ContosoPermit.Scripts.PermitForm = {
8
9     handleOnLoad: function (executionContext) {
10         console.log('on load - permit form');
11     },
12
13     __namespace: true
14 }
15

```

2. Add a function to OnChange permit type event

- Add the function mentioned below to the **PermitFormFuntions** file inside the function created in Step 1(d). Once this is done, click **File** and **Save All**.

```

handleOnChangePermitType: function (executionContext) {

    console.log('on change - permit type');

},

```

```

7 ContosoPermit.Scripts.PermitForm = {
8
9     handleOnLoad: function (executionContext) {
10         console.log('on load - permit form');
11     },
12
13     handleOnChangePermitType: function (executionContext) {
14         console.log('on change - permit type');
15     },
16
17     __namespace: true
18 }
19

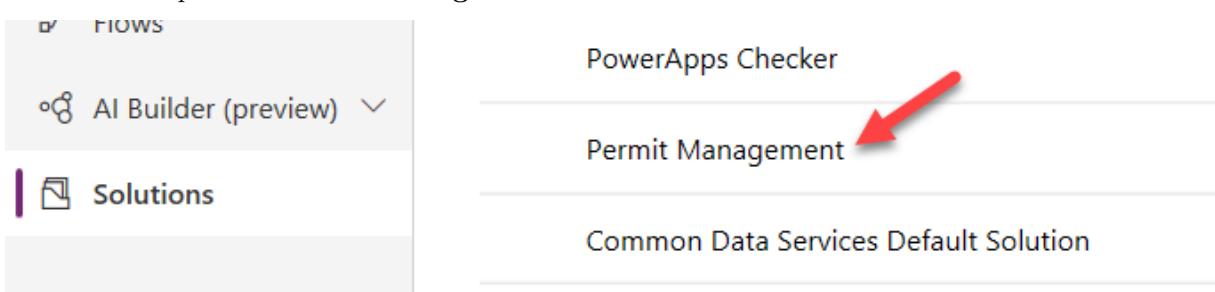
```

18.3 Task #3: Load Web Resources

In this task, you will upload the JavaScript files as web resources. Here, you will also edit the Permit Table main form and associate the new web resource with its form. Finally, you will register your functions to be called on specific form events.

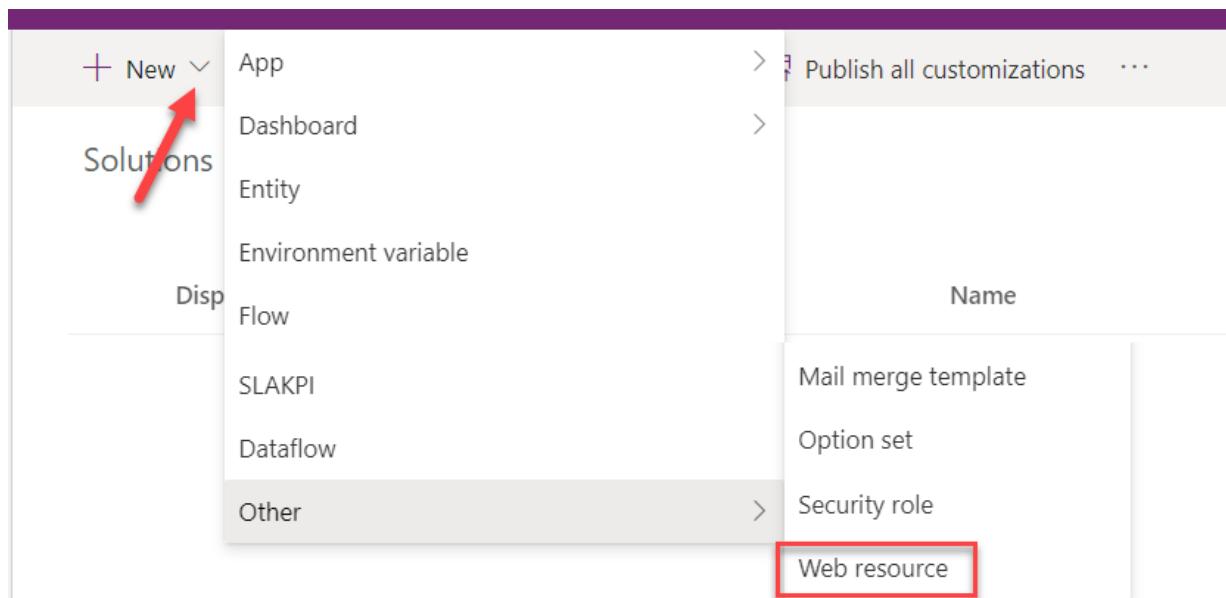
1. Open the Permit Management solution

- Sign in to [Power Apps maker portal](#).
- Select your **Dev** environment.
- Select **Solutions**.
- Click to open the **Permit Management** solution.



2. Add web resource to the solution

- Click **+ New**.
- Click **Other** and select **Web Resources**.

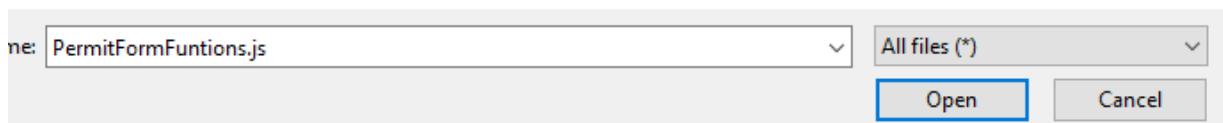


- Enter **PermitFormScripts.js** for **Name**.
- Enter **Permit Form Scripts** for **Display Name**.
- Select **Script (Jscript)** for **Type**.
- Select **English** for **Language**.
- Click **Browse**.

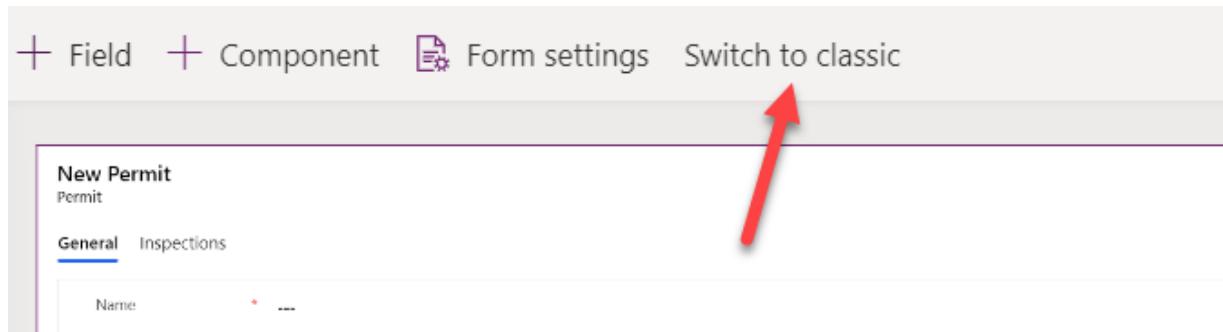
The screenshot shows the 'Web Resource: New' configuration page. The 'General' tab is selected. The 'Name' field contains 'contoso_PermitFormScripts.js' and the 'Display Name' field contains 'Permit Form Scripts'. In the 'Content' section, the 'Type' is set to 'Script (JScript)' and the 'Language' is set to 'English'. A red arrow points to the 'Browse...' button next to the 'Upload File' input field. The 'URL' field is empty.

- Select the **PermitFormFunctions.js** file and click **Open**.

Name	Date modified	Type	Size
PermitFormFuntions.js	8/28/2019 9:43 AM	JavaScript File	1 KB

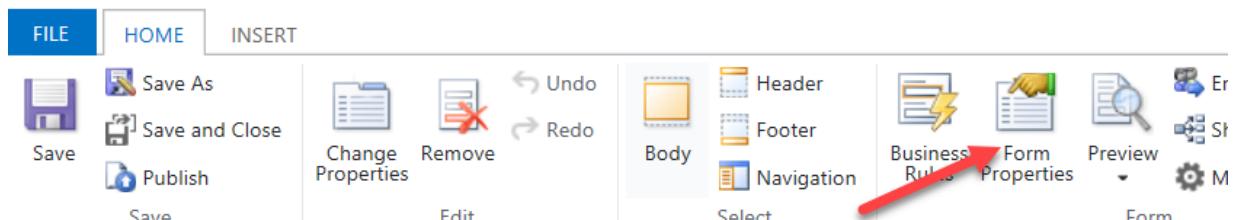


- Click **Save** and wait until the changes are saved.
 - Click **Publish** and wait for the publishing to complete.
 - Close the web resource editor Window.
 - Click **Done**.
3. Open the Permit main form.
- Make sure you are still in the solution.
 - Click to open the **Permit** Table.
 - Select the **Forms** tab and click to open the **Main** form.
4. Switch to Classic.
- Note:** Currently, you must switch to the classic solution explorer to complete the hookup of the client script to the form. In the future, this will be moved to make.powerapps.com and will not require switching to the classic tools.
- Click on the **Switch to Classic**. This will open the classic UI in a new browser window.



5. Add the script to the permit form

- Click **Form Properties**.



- Click **Add Library**.

Event List

Form Libraries

Manage libraries that will be available in the form.



Add



Remove



Up



Down



Edit

Name

Display Name

- Search for **contoso**, select **PermitFormScripts** entry with Display Name as **PermitFormScripts** and click **Add**.

LOOK TO: **WEB RESOURCE**

Look in: **WebResource Lookup View Fc**

Search: **contoso**

Name	Display Name...	Language	...
contoso_PermitFormScripts.js	Permit Form...	English(1033)	

1 - 1 of 1 (1 selected) Page 1

New

Add

Cancel

Remove Value

- Add OnLoad event handler.

- Go to the **Event Handlers** section.
- Select **Form** from **Control Dropdown**.
- Select **OnLoad** from **Event Dropdown**.
- Click **Add**.

Event Handlers

Manage functions that are called for form or field events.

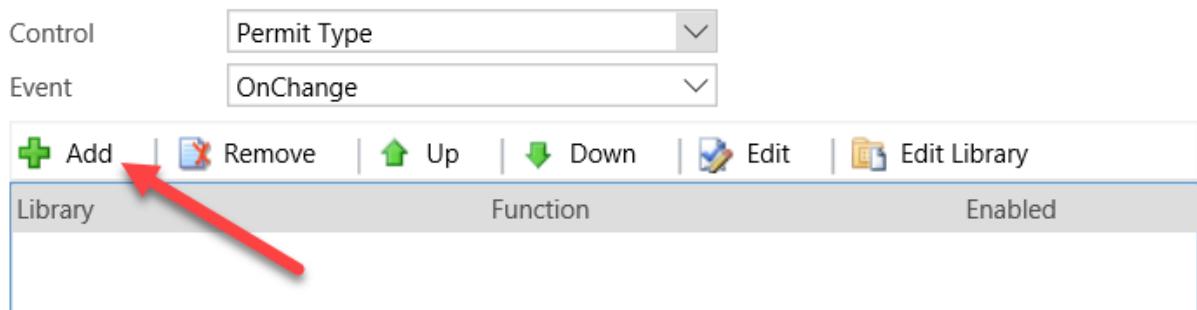
Control	Form	
Event	OnLoad	
Add Remove Up Down Edit Edit Library		
Library	Function	Enabled

- Select **Contoso_**PermitFormScripts.js in the dropdown for **Library**.
- Enter **ContosoPermit.Scripts.PermitForm.handleOnLoad** in the textbox for **Function**.
- Check the **Pass Execution Context** checkbox.
- Click **OK**.

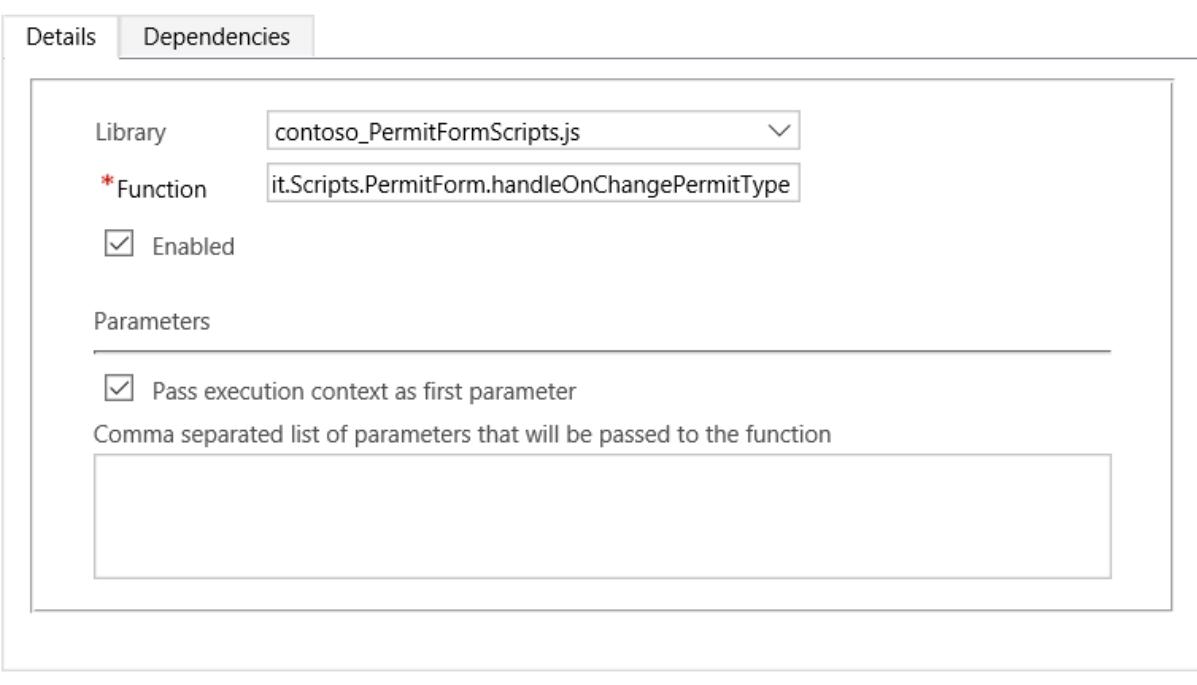
Details	Dependencies														
<table border="1"><tr><td>Library</td><td>contoso_PermitFormScripts.js</td></tr><tr><td>*Function</td><td>ContosoPermit.Scripts.PermitForm.handleOnLoad</td></tr><tr><td><input checked="" type="checkbox"/> Enabled</td><td></td></tr><tr><td colspan="2">Parameters</td></tr><tr><td colspan="2"><input checked="" type="checkbox"/> Pass execution context as first parameter</td></tr><tr><td colspan="2">Comma separated list of parameters that will be passed to the function</td></tr><tr><td colspan="2"><input type="button" value="OK"/> <input type="button" value="Cancel"/></td></tr></table>	Library	contoso_PermitFormScripts.js	*Function	ContosoPermit.Scripts.PermitForm.handleOnLoad	<input checked="" type="checkbox"/> Enabled		Parameters		<input checked="" type="checkbox"/> Pass execution context as first parameter		Comma separated list of parameters that will be passed to the function		<input type="button" value="OK"/> <input type="button" value="Cancel"/>		
Library	contoso_PermitFormScripts.js														
*Function	ContosoPermit.Scripts.PermitForm.handleOnLoad														
<input checked="" type="checkbox"/> Enabled															
Parameters															
<input checked="" type="checkbox"/> Pass execution context as first parameter															
Comma separated list of parameters that will be passed to the function															
<input type="button" value="OK"/> <input type="button" value="Cancel"/>															

7. Add Permit Type OnChange event handler.

- Go to the **Event Handlers** section.
- Select **Permit Type** from the dropdown for **Control**.
- Select **OnChange** from the dropdown for **Event**.
- Click **Add**.

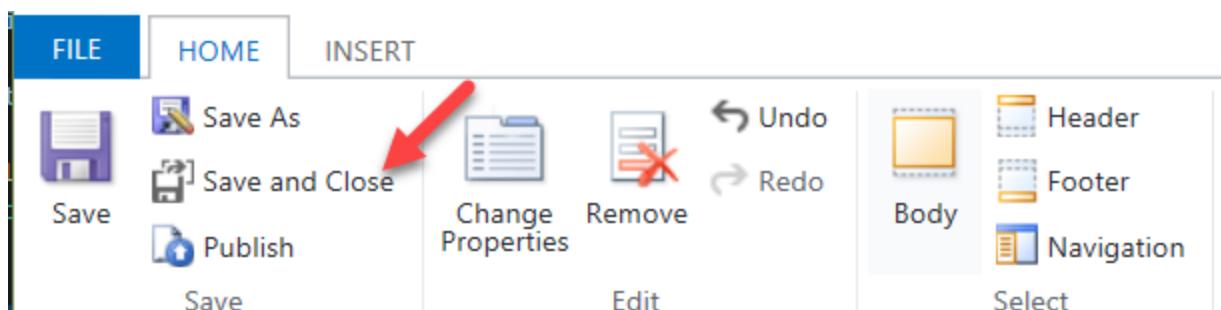


- Select **Contoso_PermitFormScripts.js** for **Library**.
- Enter **ContosoPermit.Scripts.PermitForm.handleOnChangePermitType** in the textbox for **Function**.
- Check the **Pass Execution Context** checkbox.
- Click **OK**.



8. Save and publish your changes

- Click **OK** to close the **Form Properties** window.
- Click **Save and Close**.



- Click on the <- Back button.

• Go back to the solution by clicking on the solution name.

Solutions > Permit Management > Permit

Fields Relationships Business rules Views Forms Dashl

• Click **Publish All Customizations** and wait for the publishing to complete.

+ New Add existing Delete Export Publish all customizations ...

DO NOT navigate away from this page

18.4 Task #4: Test Event Handlers

In this task, you will be doing a test to ensure that you have correctly hooked up your functions to the permit form events.

1. Start the Permit Management application

- Select **Apps**.
- Click to start the **Permit Management** application and this will open the app in New Window.

Apps in Rage Dev (orga5253b09)

Recent apps Shared with me Apps I can edit Org apps

Name	Modified
Inspector	1 day ago
Solution Health Hub	2 days ago
Permit Management	4 days ago

2. Open a Permit record

- Select **Permits** from the Site Map.
- Click to open a permit record.

Home Recent Pinned

Permits

Permits

Active Permits

Name

Test Permit

3. Open Edge Dev Tools

- Press **F12** or right click and select **Inspect**.
- Select the **Console** from top menu and clear console.

Elements **Console** Sources Network Performance Memory Ap

top Filter Default levels ▾

4. Refresh and confirm the OnLoad event handler function runs

- Go to the **Permit** record and click **Refresh**.

New Deactivate Delete Refresh Assign Share Email

PERMIT
Test Permit

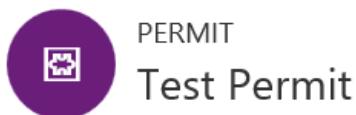
- Go to the **Dev Tools** and you should now be able to see the **on load – permit form** message.

The screenshot shows the Chrome Dev Tools interface with the Network tab selected. A log entry is visible in the logs section:

```
on load - permit form
```

- Remove Permit Type and confirm the OnChange Permit Type event handler function runs

- Go to the **Permit** record and remove the **Permit Type**.



General Inspections Related

Name	* Test Permit
Permit Type	New Construction X
Build Site	One Microsoft Way
Contact	John Doe

- Go to the **Dev Tools** and you should now be able to see the **on change – permit type** message.

The screenshot shows the Chrome Dev Tools interface with the Network tab selected. Log entries are visible in the logs section:

```
on load - permit form
on change - permit type
```

- Close the **Dev Tools**.

19 Exercise #2: Show and Hide Tabs

Objective: In this exercise, you will create a script that will show and hide the inspections tab based on the permit type Table's "required inspections" Column value.

19.1 Task #1: Create Function

- Create a function that will run when the Permit form loads and when the Permit Type value changes

- Go back to **Visual Studio Code**.
- Add the function mentioned below to **PermitFormFuntions** inside the **PermitForm** function.

```
_handlePermitTypeSettings: function (executionContext) {
```

```

14     console.log('on change - permit type');
15   },
16
17   _handlePermitTypeSettings: function (executionContext) {
18
19   },
20   _namespace: true
21 }
22
23

```

2. Get form context from the execution context

- Add the script mentioned below inside **_handlePermitTypeSettings** function.

```
var formContext = executionContext.getFormContext();
```

3. Get the Permit Type value from the form.

- Add the script mentioned below inside the **_handlePermitTypeSettings** function. **contoso_permittype** is the logical name of the Permit Type Column. You can verify this in the Table metadata.

```
var permitType = formContext.getAttribute("contoso_permittype").getValue();
```

4. Check if the Permit Type has value.

- Add the script mentioned below inside the **_handlePermitTypeSettings** function.

```
if (permitType == null) {
} else {
}
```

```

_handlePermitTypeSettings: function (executionContext) {
  var formContext = executionContext.getFormContext();
  var permitType = formContext.getAttribute("contoso_permittype").getValue();
  if (permitType == null) {

  } else {
  }
},
_namespace: true

```

5. Hide the Inspections tab and return if Permit type is null.

- Add the script mentioned below inside the **_handlePermitTypeSettings** function. **inspectionsTab** is the name of the Inspections tab (This is configured while configuration of the Model Driven App in Lab1 Module 2).

```
formContext.ui.tabs.get("inspectionsTab").setVisible(false);
return;
```

```
    _onControlChange("permitType");
},
_handlePermitTypeSettings: function (executionContext) {
    var formContext = executionContext.getFormContext();
    var permitType = formContext.getAttribute("contoso_permittype").getValue();
    if (permitType == null) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        return;
    } else {
    }
},
namespace: true
```

19.2 Task #2: Get Inspection Type Record

In this task, you will use the web API to retrieve the permit type lookup record associated with the current permit record that is currently displayed in the form.

- #### 1. Get the Permit Type ID

- Add the script mentioned below in the else statement of the `_handlePermitTypeSettings` function.

```
var permitTypeID = permitType[0].id;
```

2. Retrieve the Permit Type record and show alert if there are errors

- Add the script mentioned below in the else statement of the `_handlePermitTypeSettings` function. `contoso_permittype` is the logical name of the Permit Type Table.

```
Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {  
},
```

```
function (error) { alert('Error:' + error.message) };
```

```
    console.log(`on change - permit type`);
},
_handlePermitTypeSettings: function (executionContext) {
    var formContext = executionContext.getFormContext();
    var permitType = formContext.getAttribute("contoso_permittype").getValue();
    if (permitType == null) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        return;
    } else {
        var permitTypeID = permitType[0].id;
        Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
            },
            function (error) { alert('Error:' + error.message) });
    }
},
});
```

3. Check if “Require Inspections” Column value is true

- Add the script mentioned below in the **retrieveRecord** function call. contoso_requireinspections is the logical name of the Require Inspections Column of the Permit Type Table.

```
if (result contoso requireinspections) {
```

```

    } else {
}

```

4. Make the Inspections tab visible if Require Inspections is true

- Add the script mentioned below in the if statement of the **retrieveRecord** call.

```
formContext.ui.tabs.get("inspectionsTab").setVisible(true);
```

5. Hide the Inspections tab if Require Inspections is not true

- Add the script mentioned below in the else statement of the **retrieveRecord** call.

```
formContext.ui.tabs.get("inspectionsTab").setVisible(false);
```

```

_handlePermitTypeSettings: function (executionContext) {
    var formContext = executionContext.getFormContext();
    var permitType = formContext.getAttribute("contoso_permittype").getValue();
    if (permitType == null) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        return;
    } else {
        var permitTypeID = permitType[0].id;
        Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
            if (result.contoso_requireinspections) {
                formContext.ui.tabs.get("inspectionsTab").setVisible(true);
            } else {
                formContext.ui.tabs.get("inspectionsTab").setVisible(false);
            }
        },
        function (error) { alert('Error: ' + error.message) });
    }
},

```

6. Call the **_handlePermitTypeSettings** function from the **handleOnLoad** function.

- Go to the **handleOnLoad** function and add the script mentioned below.

```
ContosoPermit.Scripts.PermitForm._handlePermitTypeSettings(executionContext);
```

7. Call the **_handlePermitTypeSettings** function from the **handleOnChangePermitType** function.

- Go to the **handleOnChangePermitType** function and add the script mentioned below.

```
ContosoPermit.Scripts.PermitForm._handlePermitTypeSettings(executionContext);
```

```

ContosoPermit.Scripts.PermitForm = {

    handleOnLoad: function (executionContext) {
        ContosoPermit.Scripts.PermitForm._handlePermitTypeSettings(executionContext);
        console.log('on load - permit form');
    },

    handleChangePermitType: function (executionContext) {
        ContosoPermit.Scripts.PermitForm._handlePermitTypeSettings(executionContext);
        console.log('on change - permit type');
    },

    handlePermitTypeSettings: function (executionContext) {

```

Click File and **Save All**.

19.3 Task #3: Load Updated Script

1. Open the Permit Form Script web resource.
 - Navigate to [Power Apps maker portal](#).
 - Select your **Dev** environment.
 - Select **Solutions**.
 - Click to open the **Permit Management** solution.

The screenshot shows the 'SOLUTIONS' section of the Power Apps maker portal. On the left, there's a sidebar with 'FLOWS' and 'AI Builder (preview)'. Below that is a 'Solutions' section with a purple icon. To the right, a list of solutions is shown:

- PowerApps Checker
- Permit Management** (with a red arrow pointing to it)
- Common Data Services Default Solution

- Locate and click to open the **Permit Form Script** web resource.

Solutions > **Permit Management**

Display name	Name
Contact	... contact
Permit	... contoso_permit
Permit Form Scripts	... contoso_PermitFormScripts.js
Permit Management	... contoso_PermitManagement

2. Load the updated version of permitFormFuntion.js

- Click **Choose file**.

The screenshot shows the configuration page for the 'Permit Form Scripts' solution. It has two main sections: 'General' and 'Content'.
In the 'General' section:

- Name: contoso_PermitFormScripts.js
- Display Name: Permit Form Scripts
- Description: (empty text area)

In the 'Content' section:

- Type: Script (JScript)
- Language: English
- Upload File: Choose File (with a red arrow pointing to it)

- Select **PermitFormFunctions.js** and click **Open**.

3. Save and Publish your changes

- Click **Save** and wait until the changes are saved.
- Click **Publish** and wait for the publishing to complete.

Solution: Permit Management

Web Resource: Permit Form Scripts

General Dependencies

General

Name *	contoso_	PermitFormScripts.js
Display Name	Permit Form Scripts	
Description		

DO NOT close this window. You will need to come back to this window in the next exercise.

19.4 Task #4: Test Your Changes

1. Start the Permit Management application
 - Select the browser tab or window for the Power Apps maker portal and click **Done** on the popup.

Currently editing a
Customization.Type_WebResource

When you're done editing the
Customization.Type_WebResource, click Done below to
return to the page. This will refresh the page and fetch
your changes.

Done

- Select **Apps**.
- Click to open the **Permit Management** application.

2. Open Permit record.
 - Select Permits from the Site Map.
 - Click to open a **Permit** record.

Pinned

Permits

Permits

Inspections

✓ | Name

Test Permit

3. Check if the **Permit Type** Column is empty and if it is, the **Inspections** tab is hidden. In this case, the Permit Type is null.



PERMIT
Test Permit

General	Related
----------------	----------------

Name	* Test Permit
Permit Type	---
Build Site	One Microsoft Way
Contact	John Doe

4. Select Permit Type.

- Click on the **Permit Type** lookup.
- Select **New Construction**.
- Check if the **Inspections** tab is still hidden. If so, in this case, the Require Inspections Column value is false/No



PERMIT
Test Permit

General	Related
----------------	----------------

Name	* Test Permit
Permit Type	New Construction
Build Site	One Microsoft Way

5. Set **Require Inspections** Column value of the **Permit Type** to Yes.

- Click on the selected **Permit Type**.



PERMIT
Test Permit

General Related

Name	* Test Permit
Permit Type	New Construction (arrow pointing to this field)
Build Site	One Microsoft Way
Contact	

- Set the **Require Inspections** to Yes.



PERMIT TYPE
New Construction

General Related

Name	* New Construction
Require Inspections	Yes (red box around this field)
Require Size	No

- Click **Save** button on the bottom right of the screen.
 - Click on the browser back button.
6. You should now be able to see the Inspections tab.
- Select the **Inspections** tab.

PERMIT

Test Permit

General Inspections Related

Name	*	Test Permit
Permit Type		New Construction

- The user should now be able to view/add inspections to the sub-grid.

PERMIT	Status Reason
Test Permit	Active

General Inspections Related

Inspections

+ Add New Inspection ...

Search for records

Name	Inspection Type	Scheduled Date	Sequence
Framing Inspection	Initial Inspection	8/30/2019	---

20 Exercise #3: Toggle *required property on the Columns

Objective: In this exercise, you will create a script that will make the “New Size” Column required when the “Require Size” Column value is set to Yes. If the “Require Size” Column value is set to No, remove the requirement and make it optional. You will also hide the “New Size” Column. This logic will be driven by a Column on the permit type record that was retrieved using web API in the previous exercise.

20.1 Task #1: Create Function

- Locate the `_handlePermitTypeSettings` function
 - Go back to **Visual Studio Code**.
 - Locate the `_handlePermitTypeSettings` function.
- If `permitType` is null, remove the requirement and hide the “New Size” Column.
 - Add the script mentioned below in the `if permitType == null` statement. `contoso_newsize` is the logical name of the New Size Column.

```
formContext.getAttribute("contoso_newsize").setRequiredLevel("none");

formContext.ui.controls.get("contoso_newsize").setVisible(false);
```

```

},
_handlePermitTypeSettings: function (executionContext) {
    var formContext = executionContext.getFormContext();
    var permitType = formContext.getAttribute("contoso_permittype").getValue();
    if (permitType == null) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        formContext.getAttribute("contoso_newsize").setRequiredLevel("none");
        formContext.ui.controls.get("contoso_newsize").setVisible(false);
        return;
    } else {

```

3. Check if “Require Size” Column value of the Permit Type is set to Yes

- Add the script mentioned below inside the retrieveRecord function.

```

        if (result.contoso_requiresize) {

    } else {

        var permitTypeID = permitType[0].id;
        Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
            if (result.contoso_requireinspections) {
                formContext.ui.tabs.get("inspectionsTab").setVisible(true);
            } else {
                formContext.ui.tabs.get("inspectionsTab").setVisible(false);
            }

            if (result.contoso_requiresize) {

        } else {

    }

},
function (error) { alert('Error:' + error.message) });

```

4. If “Require Size” Column value of the Permit Type is set to Yes, make the “New Size” Column visible and as required.

- Add the script mentioned below in the `if result.contoso_requiresize` statement. `contoso_requiresize` is the logical name of the Require Size Column.

```

        formContext.ui.controls.get("contoso_newsize").setVisible(true);

        formContext.getAttribute("contoso_newsize").setRequiredLevel("required");

```

```

} else {
    var permitTypeID = permitType[0].id;
    Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
        if (result.contoso_requireinspections) {
            formContext.ui.tabs.get("inspectionsTab").setVisible(true);
        } else {
            formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        }

        if (result.contoso_requiresize) {
            formContext.ui.controls.get("contoso_newsize").setVisible(true);
            formContext.getAttribute("contoso_newsize").setRequiredLevel("required");
        } else {
        }
    },

```

5. If Require Size Column value of the Permit Type is not set to Yes, remove the “New Size” Column not required and hide it.

- Add the script mentioned below inside the else statement.

```

formContext.getAttribute("contoso_newsize").setRequiredLevel("none");

formContext.ui.controls.get("contoso_newsize").setVisible(false);

Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
    if (result.contoso_requireinspections) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(true);
    } else {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
    }

    if (result.contoso_requiresize) {
        formContext.ui.controls.get("contoso_newsize").setVisible(true);
        formContext.getAttribute("contoso_newsize").setRequiredLevel("required");
    } else {
        formContext.getAttribute("contoso_newsize").setRequiredLevel("none");
        formContext.ui.controls.get("contoso_newsize").setVisible(false);
    }
}

```

6. The `_handlePermitTypeSettings` function should now look like the image below.

```

_handlePermitTypeSettings: function (executionContext) {
    var formContext = executionContext.getFormContext();
    var permitType = formContext.getAttribute("contoso_permittype").getValue();
    if (permitType == null) {
        formContext.ui.tabs.get("inspectionsTab").setVisible(false);
        formContext.getAttribute("contoso_newsize").setRequiredLevel("none");
        formContext.ui.controls.get("contoso_newsize").setVisible(false);
        return;
    } else {
        var permitTypeID = permitType[0].id;
        Xrm.WebApi.retrieveRecord("contoso_permittype", permitTypeID).then(function (result) {
            if (result.contoso_requireinspections) {
                formContext.ui.tabs.get("inspectionsTab").setVisible(true);
            } else {
                formContext.ui.tabs.get("inspectionsTab").setVisible(false);
            }

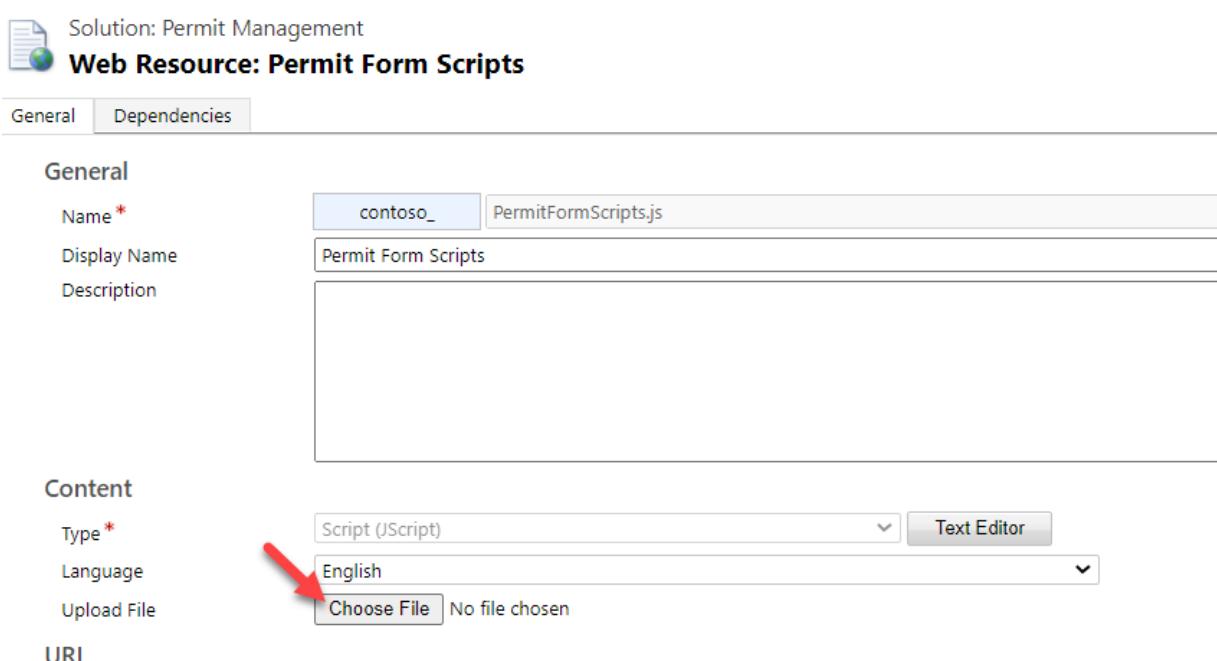
            if (result.contoso_requiresize) {
                formContext.ui.controls.get("contoso_newsize").setVisible(true);
                formContext.getAttribute("contoso_newsize").setRequiredLevel("required");
            } else {
                formContext.getAttribute("contoso_newsize").setRequiredLevel("none");
                formContext.ui.controls.get("contoso_newsize").setVisible(false);
            }
        },
        function (error) { alert('Error: ' + error.message) });
    }
}

```

Click File and Save All.

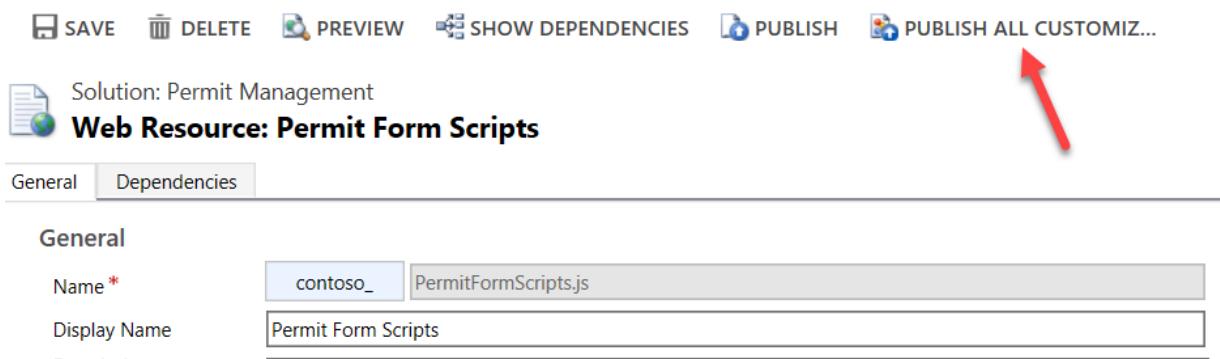
20.2 Task #2: Load Updated Script

1. Open the Permit Form Script web resource.
 - Go back to the Permit Form Scripts web resource.
 - Click **Choose File**.



- Select the **PermitFormFunctions.js** you updated and click **Open**.
2. Save and Publish your changes

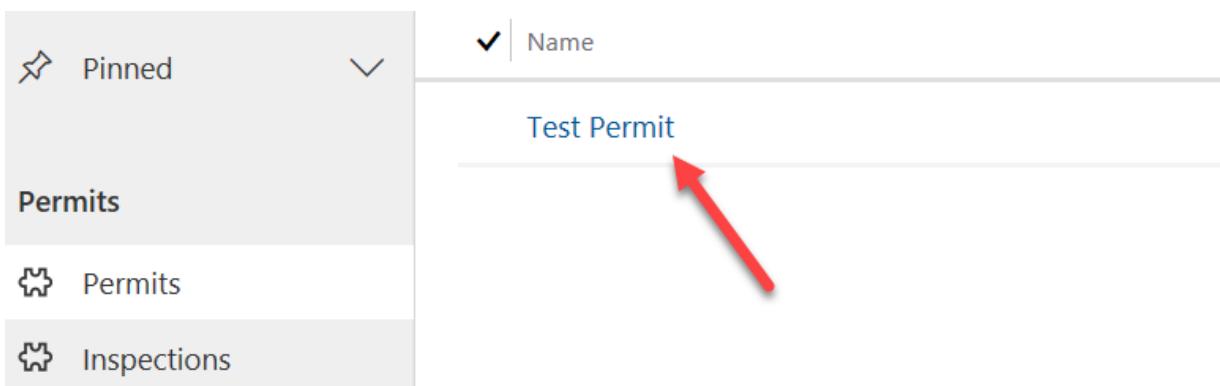
- Click **Save** and wait until the changes are saved.
- Click **Publish All Customizations** and wait for the publishing to complete.



Close the Web Resource editor.

20.3 Task #3: Test Your Changes

1. Start the Permit Management application
 - Log on to <https://make.powerapps.com> and select your **Dev** environment.
 - Select **Apps**.
 - Click to open the **Permit Management** application.
2. Open Permit record.
 - Select Permits.
 - Click to open a **Permit** record.



3. Check if the **New Size** Column is hidden. If so, then it is because in this case, the “Require Size” Column of the Permit Type is set to NO.



PERMIT
Test Permit

General Inspections Related

Name	* Test Permit
Permit Type	New Construction
Build Site	One Microsoft Way
Contact	John Doe
Start Date	* 8/30/2019

4. Set **Require Size** Column value of the **Permit Type** to Yes.

- Click on the selected **Permit Type**.



PERMIT
Test Permit

General Inspections Related

Name	* Test Permit
Permit Type	New Construction
Build Site	One Microsoft Way

- Set the **Require Size** to Yes.



PERMIT TYPE
New Construction

General Related

Name * New Construction

Require Inspections Yes

Require Size Yes

Owner * MOD Administrator

- Click **Save** on the bottom right of the screen.
 - Click on the browser back button.
5. Check if the “New Size” Column is visible and it is marked as required.
- You should now be able to see “New Size” Column on the form and this is marked as required.

Test Permit

Permit

General Inspections Related

Name * Test Permit

Permit Type New Construction

Build Site One Microsoft Way

Contact John Doe

Start Date * 12/19/2019

New Size * ---

Owner * MOD Administrator

- Remove **Permit Type**.

General Inspections Related

Name	* Test Permit
Permit Type	 New Construction 
Build Site	 One Microsoft Way
Contact	 John Doe
Start Date	* 12/19/2019
New Size	* ---
Owner	*  MOD Administrator

- Check if both the **Inspections** tab and **New Size** Column are now hidden. They should be removed as soon as the “Permit Type” is removed.



General Related

Name	* Test Permit
Permit Type	---
Build Site	 One Microsoft Way
Contact	 John Doe
Start Date	* 8/30/2019
Owner	*  MOD Administrator

21 Exercise #4: Command Button Function

Objective: In this exercise, you will download and install the Ribbon Workbench tool to edit the command bar. Through this, you will also create action, create function that will lock permits, add a button to the permit Table and call the lock permit function when the button is clicked.

21.1 Task #1: Download and Install Ribbon Workbench

Ribbon Workbench is a community tool that makes it easier to edit the command bar on a form. Alternatively, you can do this without the tool, by directly editing the RibbonDiffXml.

- Get Ribbon Workbench download link

- Log on to [http://www.develop1.net/public/Download%20Ribbon%20Workbench%202013.aspx](<http://www.develop1.net/public/Download%20Ribbon%20Workbench%202013.aspx>)
- Click **Download**.

By installing the Ribbon Workbench you'll quickly be performing customizations previously only possible by time consuming and error-prone manual editi

DOWNLOAD

LEARN MORE ▾

- Provide a name and an email and the download link will be sent to that email.
- Select **Dynamics 365/Power Apps** for **Version** available and click **Send Download Link**.

Your name:

Email to send download link to:

Version:

Dynamics 365/Power Apps



By using the Ribbon Workbench I am agreeing to the [license agreement](#)

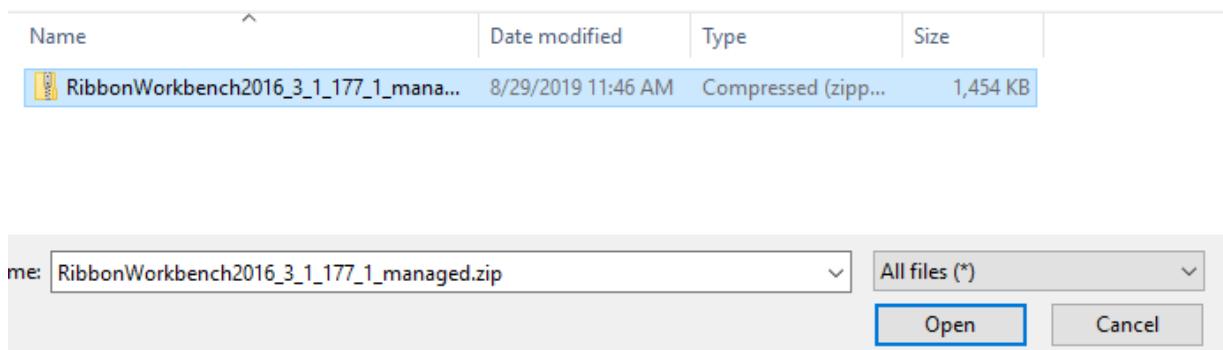
SEND DOWNLOAD LINK

- Wait to receive the download link in an email to the provided email address.
2. Download Ribbon Workbench
 - After you receive the email, click on the download button.
 - Save the downloaded solution on your machine.
 3. Install Ribbon Workbench solution.
 - Sign in to [Power Apps maker portal](#) and select your **Dev** environment.
 - Select **Solutions** and click **Import**.



Solutions		
Display name	Created ↓	Version
RibbonWorkbench2016_3_1_177_1_managed.zip	8/29/2019 11:46 AM	Compressed (zipp...) 1,454 KB

- Click **Choose File**.
- Select the **RibbonWorkbench** solution you just downloaded and click **Open**.



- Click **Next**.
- Click **Next** again.
- Click **Import** and wait for the import to complete.
- Click **Close**.

i The import of solution: Ribbon Workbench 2016 completed successfully.

Date	Type	Display Name...	Name
18:13:22.68	Process activ...	CustomiseRib...	CustomiseRib...
18:13:12.71	Process	CustomiseRib...	CustomiseRib...

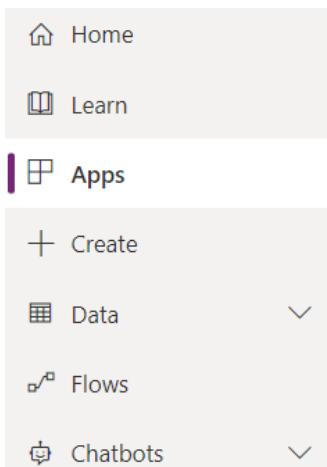
Page 1

4. Confirm the Ribbon Workbench was installed

- Select **Apps** and click to open the **Permit Management** application.

[Download Log File](#)

[Close](#)



Apps

Apps Component libraries (preview)

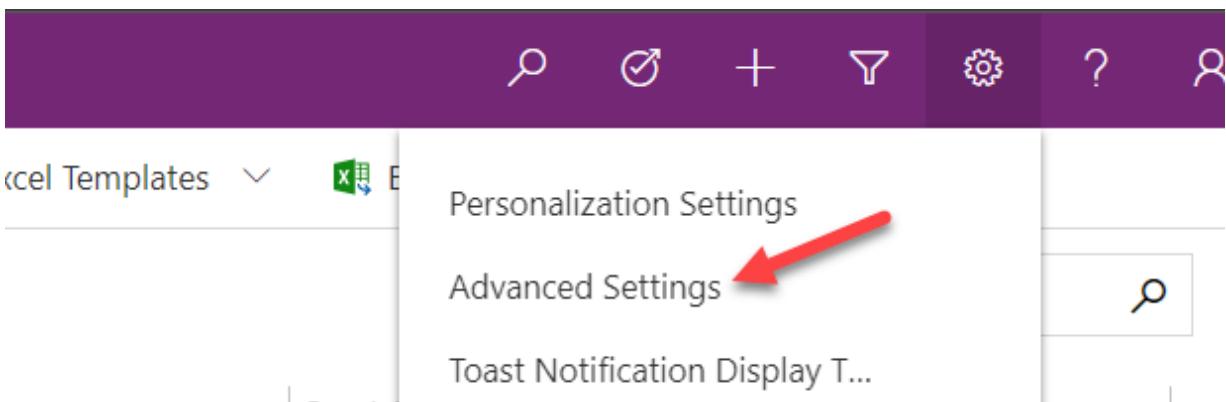
Name

Inspector

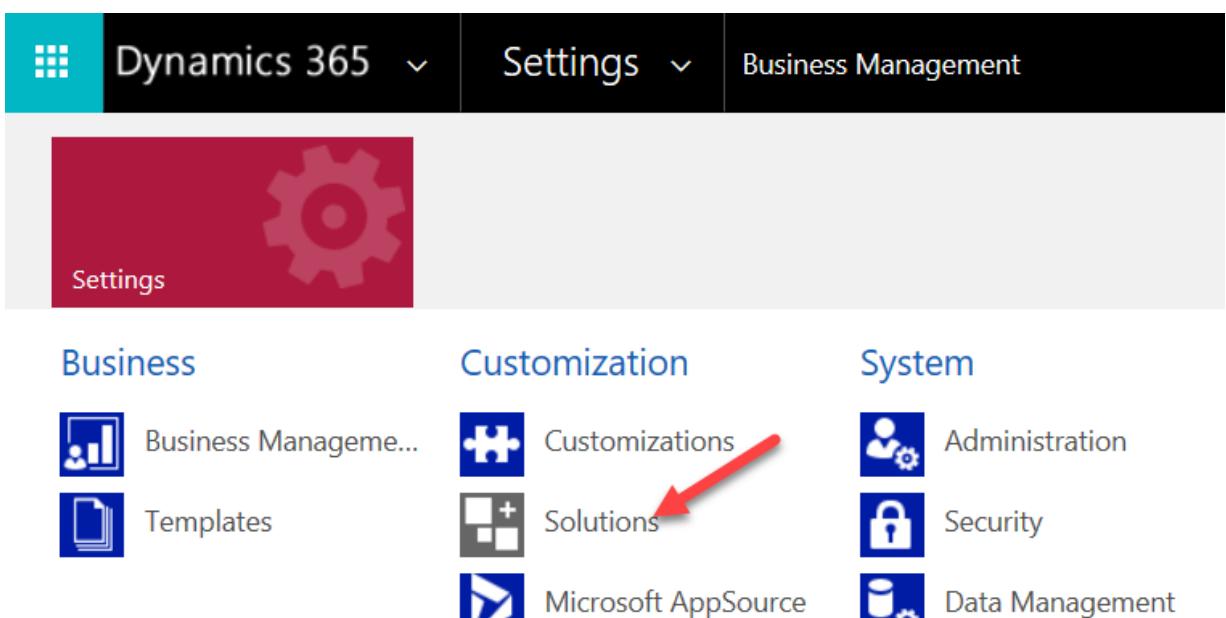
Permit Management

Solution Health Hub

- Click **Settings** and select **Advanced Settings**.



- Click **Settings | Solutions**.



- You should see the Ribbon Workbench button on the top.

5. Close the **Advanced Settings** browser tab or window.

6. Close the Permit Management application.

21.2 Task #2: Create Action Process

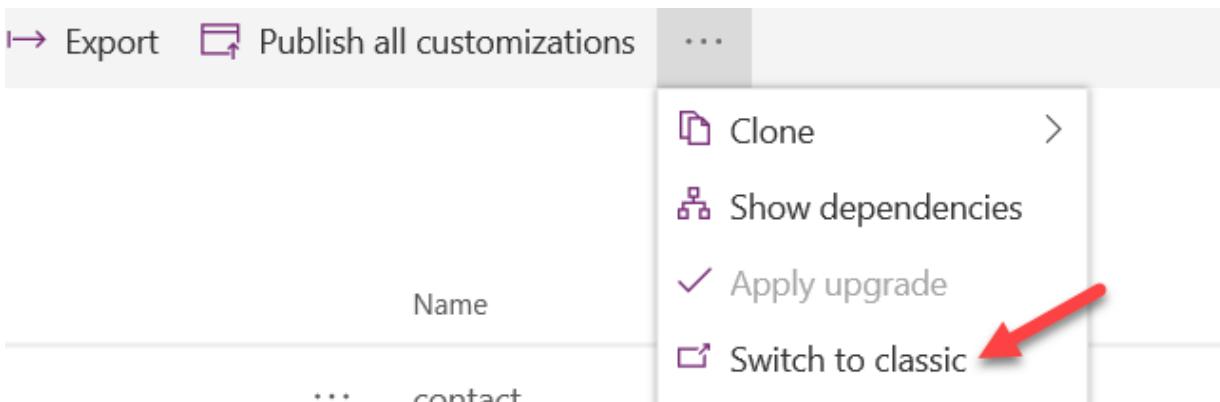
In this task, you will create a custom action that will be called to lock the permit. You will not be implementing the business logic in this lab to lock the permit. It will be completed later in the class when you build the plug-in that registers on the custom actions you will define here.

1. Open the Permit Management Solution.

- Navigate to [Power Apps makes portal](#) and make sure you have the **Dev** environment selected.
- Select **Solutions** and click to open the **Permit Management** solution.

2. Switch to Classic

- Click on the button and select **Switch to Classic**.



3. Create new process

- Select **Processes** and click **New**.

- Enter **Lock Permit** for Name, select **Action** from the dropdown for Category, select **Permit** from the dropdown for Table, select **New Blank Process** for Type, and click **OK**.

Create Process

Define a new process, or create one from an existing template. You can create four kinds of processes: business process flows, actions, dialogs, and workflows.

Process name: *	Lock Permit								
Category: *	Action	Entity: *	Permit						
Type:	<input checked="" type="radio"/> New blank process <input type="radio"/> New process from an existing template (select from list):								
<table border="1"> <thead> <tr> <th>Template Name ↑</th> <th>Primary Entity</th> <th>Owner</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td></td> </tr> </tbody> </table>				Template Name ↑	Primary Entity	Owner			
Template Name ↑	Primary Entity	Owner							
<input type="button" value="Properties"/>									
<input type="button" value="OK"/> <input type="button" value="Cancel"/>									

4. Add process arguments

- Go to the **Process Arguments** section and click **Add**.

Keep logs for workflow jobs that encountered errors

▼ Hide Process Arguments

     			
Name	Type	Required	Direction

- Enter **Reason** for Name, select **Type** as **String**, and select **Input** for Direction.

Name *	Reason
Type *	String
Entity	
Required	<input type="checkbox"/>
Direction	<input checked="" type="radio"/> Input <input type="radio"/> Output
Description	New Argument

- Click **Add** again.
- Enter **CanceledInspectionsCount** for Name, select **Type** as **Integer**, and select **Output** for

Direction.

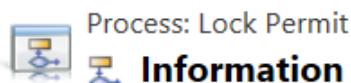
- You should now have one input and one output arguments.

▼ **Hide Process Arguments**

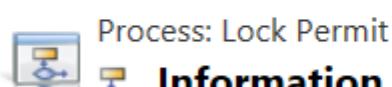
Name	Type	Required	Direction
Reason	String	Optional	Input
CanceledInspections...	Integer	Optional	Output

5. Save and activate action

- Click **Save** and wait until the changes are saved.
- Click **Activate**. This will open a pop-up for confirmation. Click **Activate**.



- Confirm the activation and wait for the action to be activated. You should now be able to see the **Deactivate** option in the Top menu.
- Click **Close**.

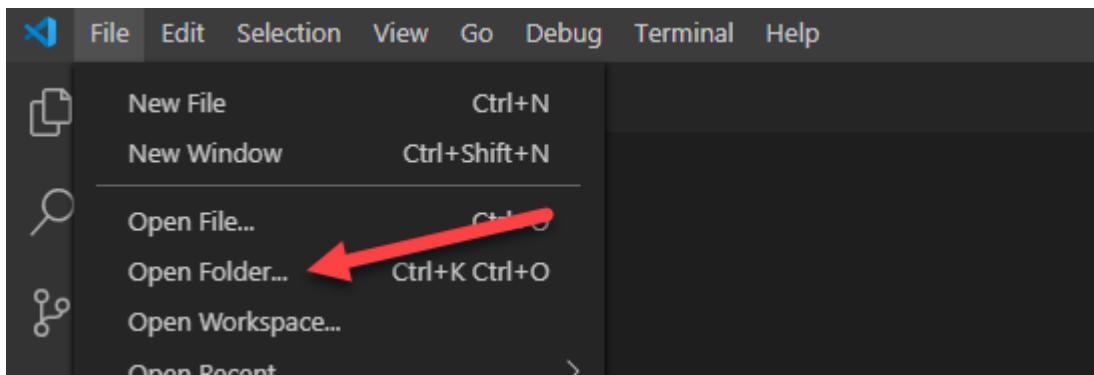


- Close the solution explorer.

21.3 Task #3: Create the Function

In this task, you will create the logic to invoke a custom action using the web API.

1. Start Visual Studio Code and open the resources you create in exercise one
 - Start **Visual Studio Code**.
 - Click **File** and select **Open Folder**.



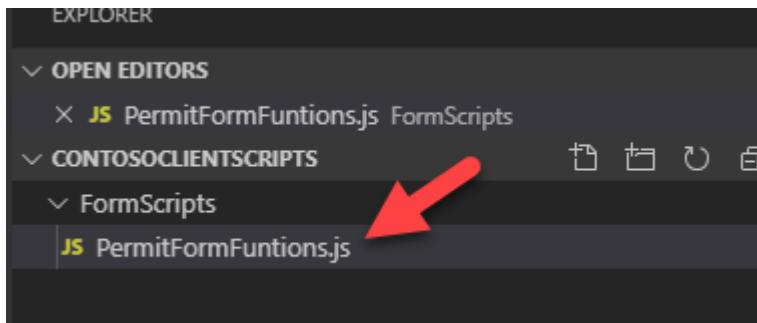
- Select the **ContosoClientScripts** folder you created in exercise one and click **Select Folder**.

Name	Date modified	Type	Size
ContosoClientScripts	8/28/2019 9:24 AM	File folder	



2. Add a function that will build the request

- Open the **PermitFormFunctions.js** file.



- Add the function below after the `_handlePermitTypeSettings` function.

```

_lockPermitRequest : function (permitID, reason) {

},
15 },
16
17 > _handlePermitTypeSettings: function (executionContext) {
47 },
48
49 _lockPermitRequest : function (permitID, reason) {
50
51 },
52
53 | __namespace: true
54

```

3. Build entity and set reason.

- Add the script mentioned below inside the `_lockPermitRequest` function.

```

this.entity = { entityType: "contoso_permit", id: permitID };

this.Reason = reason;

```

4. Build and return the request

- Add the script mentioned below in the `_lockPermitRequest` function.

```

this.getMetadata = function () {
    return {
        boundParameter: "entity", parameterTypes: {
            "entity": {
                typeName: "mscrm.contoso_permit",
                structuralProperty: 5
            },
            "Reason": {
                "typeName": "Edm.String",
                "structuralProperty": 1 // Primitive Type
            }
        },
        operationType: 0, // This is an action. Use '1' for functions and '2' for CRUD
        operationName: "contoso_LockPermit",
    };
};

```

```

_lockPermitRequest: function (permitID, reason) {
    this.entity = { entityType: "contoso_permit", id: permitID };
    this.Reason = reason;
    this.getMetadata = function () {
        return {
            boundParameter: "entity", parameterTypes: {
                "entity": {
                    typeName: "mscrm.contoso_permit",
                    structuralProperty: 5
                },
                "Reason": {
                    "typeName": "Edm.String",
                    "structuralProperty": 1 // Primitive Type
                }
            },
            operationType: 0, // This is an action. Use '1' for functions and '2' for CRUD
            operationName: "contoso_LockPermit",
        };
    };
},

```

5. Add the function that will be called from the action button.

- Add the function mentioned below after the `_lockPermitRequest` function.

```

lockPermit: function (primaryControl) {

},
66 },
67
68 lockPermit: function (primaryControl) {
69
70 },
71
72 __namespace: true
73 }

```

6. Get Permit ID and call `_lockPermitRequest`

- Get the id by adding the script mentioned below inside the `lockPermit` function.

```

formContext = primaryControl;
var PermitID = formContext.data.entity.getId().replace('{', '').replace('}', '');

```

- Call `_lockPermitRequest`. We are hardcoding the reason “Admin Lock”

```
var lockPermitRequest = new ContosoPermit.Scripts.PermitForm._lockPermitRequest(PermitID, "A
```

7. Execute the request.

- Add the script mentioned below inside the `lockPermit` function.

```
// Use the request object to execute the function
Xrm.WebApi.online.execute(lockPermitRequest).then(
    function (result) {
        if (result.ok) {
            console.log("Status: %s %s", result.status, result.statusText);
            // perform other operations as required;
            formContext.ui.setFormNotification("Status " + result.status, "INFORMATION");
        }
    },
    function (error) {
        console.log(error.message);
        // handle error conditions
    }
);

lockPermit: function (primaryControl) {
    formContext = primaryControl;
    var PermitID = formContext.data.entity.getId().replace('{', '').replace('}', '');

    var lockPermitRequest = new ContosoPermit.Scripts.PermitForm._lockPermitRequest(PermitID, "Admin Lock");

    // Use the request object to execute the function
    Xrm.WebApi.online.execute(lockPermitRequest).then(
        function (result) {
            if (result.ok) {
                console.log("Status: %s %s", result.status, result.statusText);
                // perform other operations as required;
                formContext.ui.setFormNotification("Status " + result.status, "INFORMATION");
            }
        },
        function (error) {
            console.log(error.message);
            // handle error conditions
        });
},
namespace: true
```

- Click **File** and **Save All**.

8. Load the update resource and publish.

- Log on to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select **Solutions** and click to open the **Permit Management** solution.
- Click to open the **Permit Form Scripts** web resource. This will open a new browser window.

Inspector	...	contoso_inspector_cd	Canvas App		1 h ago	-
Permit	...	contoso_permit	Entity		-	-
Permit Form Scripts	...	contoso_PermitFormS	Web Resource		In 5 h	-
Permit Management	...	contoso_PermitManag	Model-driven A		6 d ago	-

- Click **Choose File**.
- Select the **PermitFormFunctions.js** you updated and click **Open**.
- Click **Save** and wait until the changes are saved.
- Click **Publish All Customizations** and wait for the publishing to complete.

- Close the web resource editor and return to the solutions window.
- Click **Done**.

Currently editing a web resource

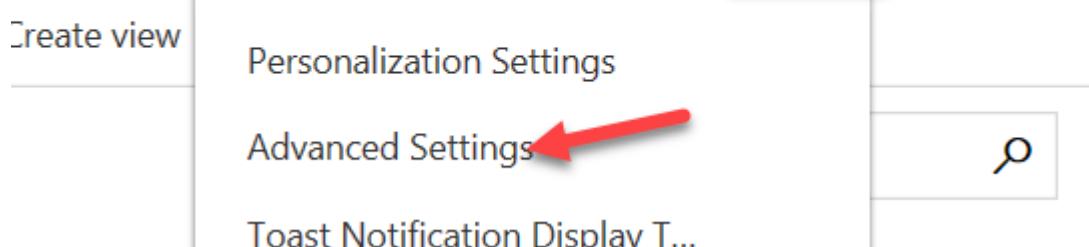
When you're done editing the web resource, click **Done** below to return to the page. This will refresh the page and fetch your changes.

Done

21.4 Task #4: Add Button to Ribbon

1. Open Ribbon Workbench

- Sign in to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select **Apps** and click to open the **Permit Management** application.
- Click **Settings** and select **Advanced Settings**.

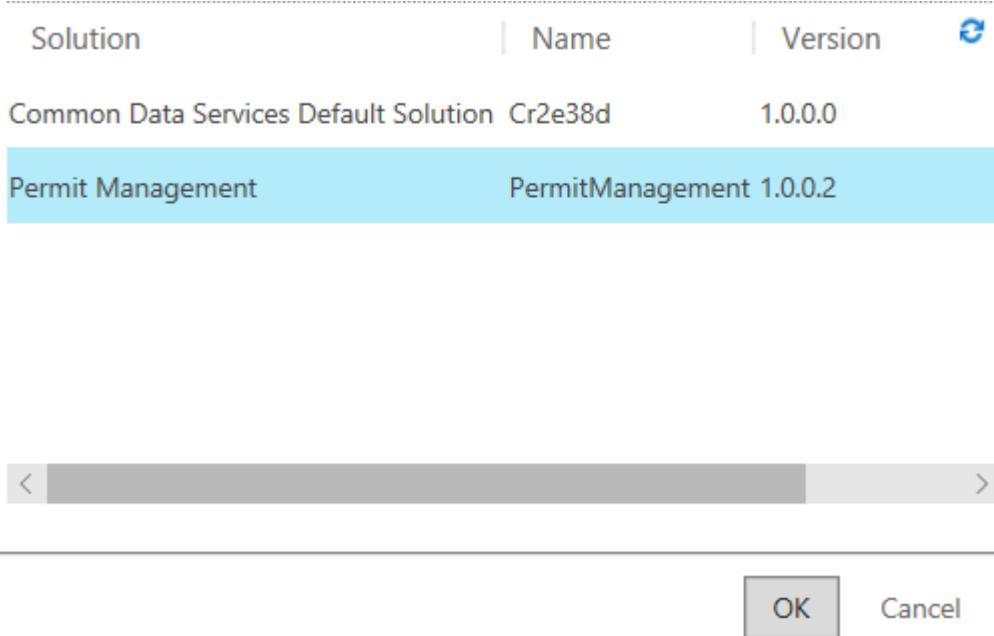


- Click **Settings | Solutions**.
- Click on the Ribbon Workbench button.

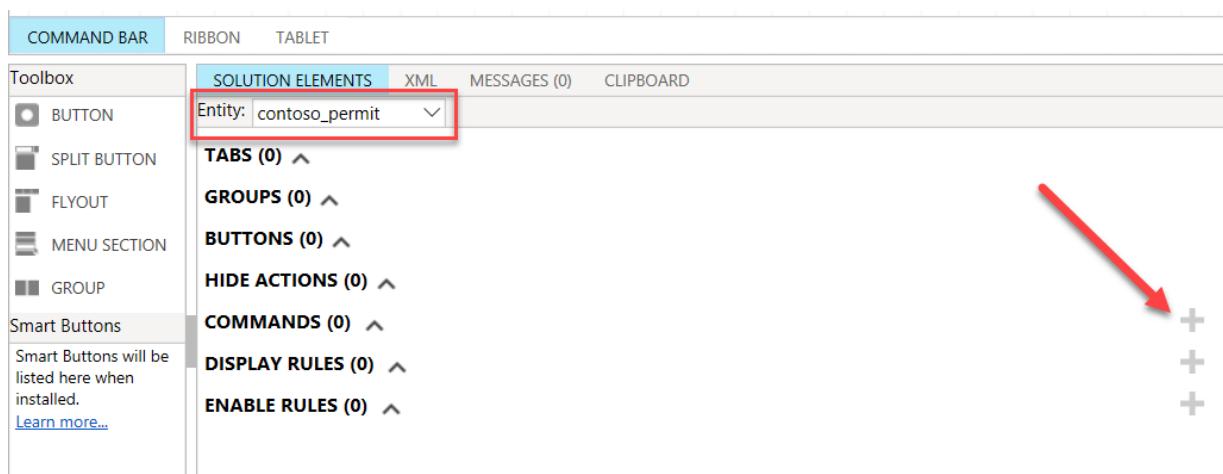
Name	Display Name	Version	Installed On

2. Load the Permit Management solution

- Select the **Permit Management** solution in the pop-up opened for selecting the Load Solution.
- Click **OK**.



- Wait for the solution to be downloaded.
3. Add command to permit Table
- Select **contoso_permit** from the dropdown for Table.
 - Click **Add Command**.



4. Add action to command
- Select the command you just created.

SOLUTION ELEMENTS XML MESSAGES (0) CLIPBOARD

Entity: contoso_permit ▾

TABS (0) ▾

GROUPS (0) ▾

BUTTONS (0) ▾

HIDE ACTIONS (0) ▾

COMMANDS (1) ▾

contoso.contoso_permit.Command0.Command

DISPLAY RULES (0) ▾

- Click Add Action.

Properties: Command

Actions ▾

Use 'Add Action' to define what this command does.

+ Add Action

Display Rules ▾

Optional: Use 'Add Display Rule' if you want to make the visibility dynamic.

Actions ▾

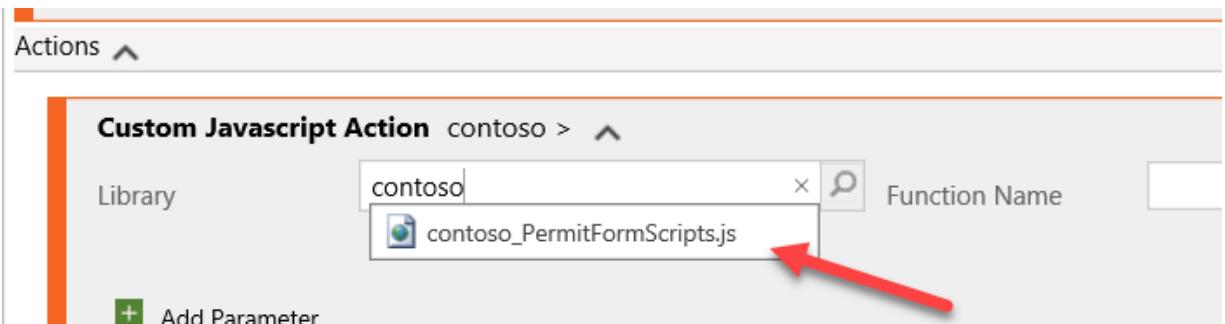
Use 'Add Action' to define what this command does.

+ Add Action

Display Rule Url Action Javascript Action

Optional: Use 'Add Display Rule' if you want to make the visibility dynamic.

- Search for **contoso** and select **contoso_PermitFormScripts.js**



- Enter **ContosoPermit.Scripts.PermitForm.lockPermit** in the textbox for **Function Name**.

Actions

Custom Javascript Action \$webresource:contoso_PermitFormScripts.js > ContosoPermit.Scripts.PermitForm.lockPermit &

Library resource:contoso_PermitFormScripts.js

Function Name >ContosoPermit.Scripts.PermitForm.lockPermit

+ Add Parameter

5. Add primary control parameter

- Click **Add Parameter** and select **CRM Parameter**.

Actions

Custom Javascript Action \$webresource:contoso_PermitFormScripts.js > ContosoPermit.Scripts.

Library \$webresource:contoso_PermitFormScript

Function Name ContosoPermit.Scripts.

+ Add Parameter

+ Add Action

Display Rules

Optional: Use 'Add Disp'

String Parmeter

Integer Parameter

Decimal Parameter

Boolean Parameter

CRM Parameter

- Select **PrimaryControl** for **Value**.

Actions

Custom Javascript Action \$webresource:contoso_PermitFormScripts.js > ContosoPermit.Scripts.PermitForm.lockPermit &

Library \$webresource:contoso_PermitFormScript

Function Name ContosoPermit.Scripts.PermitForm.lockPermit

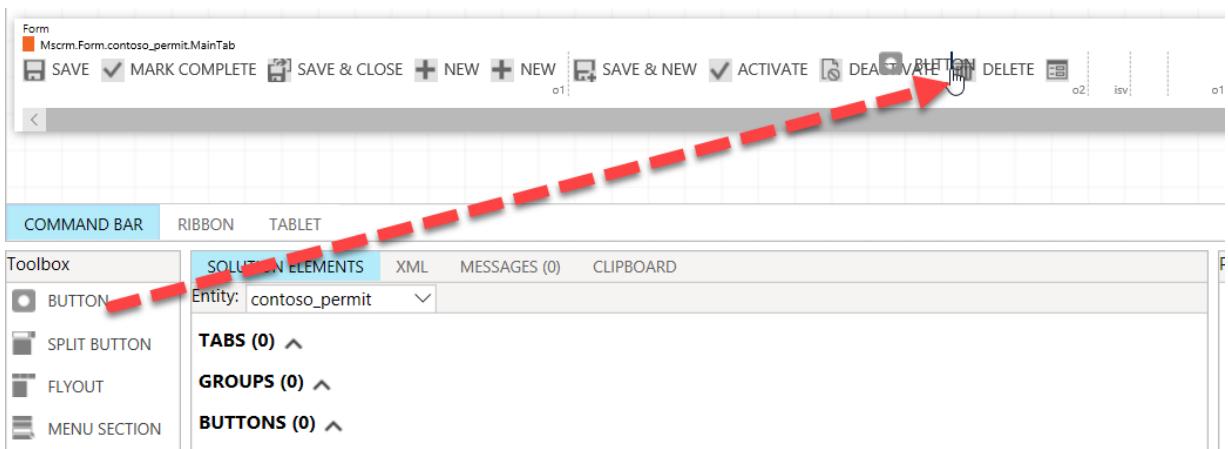
Crm Parameter = PrimaryControl

Value PrimaryControl

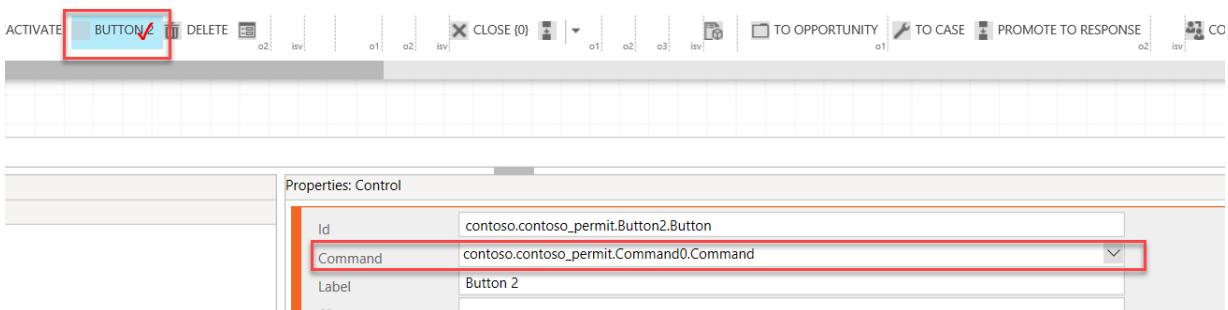
+ Add Parameter

6. Add a button select command

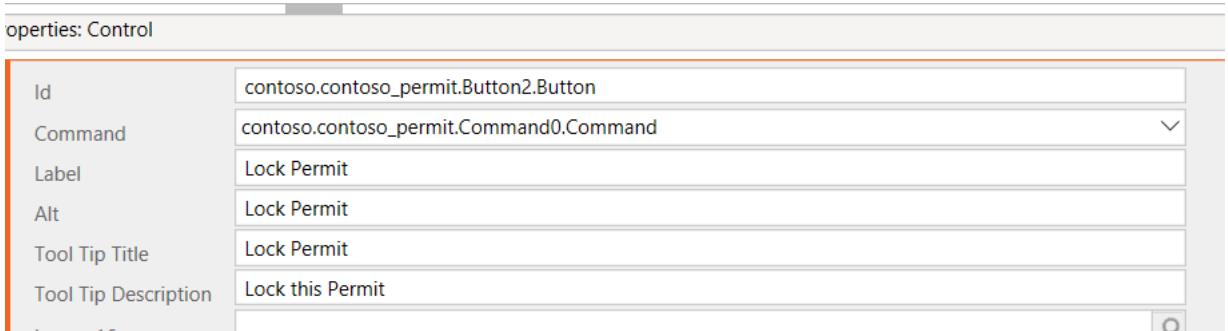
- Drag a **Button** from the **Toolbox** and drop it between **Deactivate** and **Delete**. Make sure you are adding it on the **Mscrm.Form.contoso_permit.MainTab** section.



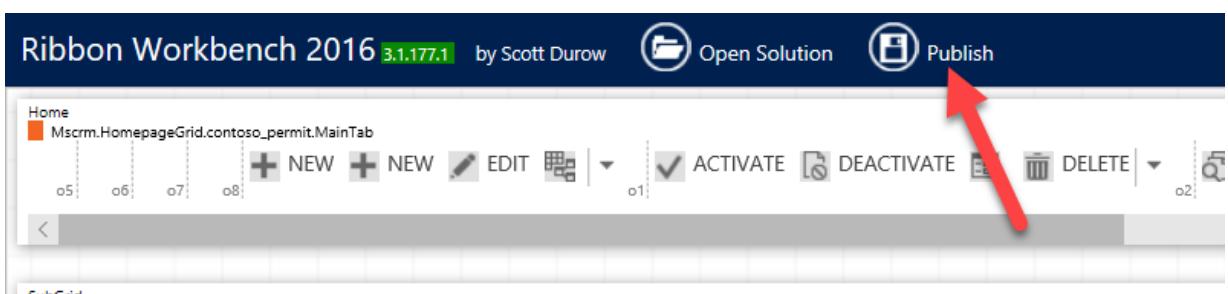
- Select the button you just added to ribbon.
- Go to the **Properties** section and select the command you created for **Command**.



- Change the **Label** value to **Lock Permit**.
- Provide **Alt** text, **Tool Tip Title**, and **Tool Tip Description**.



7. Publish your changes.



Click **OK** and wait for the publishing to complete. This might take a few minutes to complete.

21.5 Task #5: Test Command Button

1. Start the Permit Management application.
 - Log on to [Power Apps maker portal](#) and make sure you have your **Dev** environment selected.

- Select **Apps** and click to open the **Permit Management** application.

2. Open a permit record

- Select Permits.
- Click to open a permit.
- You should be able to see the button you just added.



PERMIT
Test Permit

3. Test the Command

- Click on the **Lock Permit** button.
- The script should trigger the action and you should be able to see the Status 200 notification.



PERMIT

Status Reason

21.6 lab: title: 'Lab 05: Power Apps Component Framework'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *field* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

21.7 Lab 05 – Power Apps Component Framework

22 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will develop a custom component control using the Power Apps Component Framework (PCF). This component will be used to visualize the permit inspection history timeline. As you build the component, you will see how to use prescriptive interfaces of the framework to interact with the hosting form data. To speed things up we will use a community timeline library to render the visualization. When you build such controls, you can either follow the same procedure or use popular frameworks like React or Vue to completely control the visualization that the component will render.

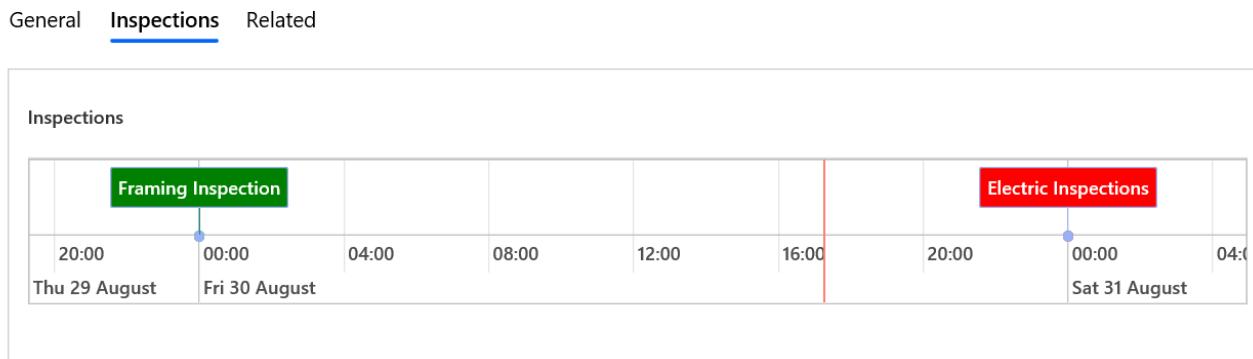
23 High-level lab steps

As part of building this component, you will complete the following steps:

- Use the Power Apps CLI to initialize the new component
- Build the component logic using Typescript
- Publish the component for use on forms

- Configure the permit form to use the component

The is what the component will look like when it is completed.



23.1 Things to consider before you begin

- What are the advantages of using a Power Apps Component Framework component over an embedded Power Apps canvas app?
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production soon.

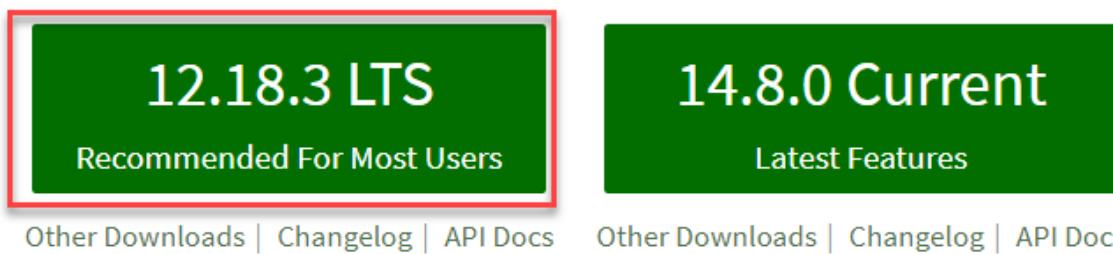
24 Exercise #1: Create the PCF Control

Objective: In this exercise, you will create a Power Apps Components Framework control using the Power Apps CLI

24.1 Task #1: Install Microsoft Power Apps CLI and Prerequisites

- Install Node.js
 - Navigate to [Node JS](#)
 - Select the latest **LTS** version.

Download for Windows (x64)



- Click **Open file**.
 - Follow the steps in setup wizard to complete installing **Node.js**
- Install .NET Framework 4.6.2 Developer Pack
 - Navigate to [Download .NET Framework 4.6.2](#)
 - Select the **Developer Pack**.

Developer Pack

The developer pack is used by software developers to create applications that run on .NET Framework, typically using Visual Studio.

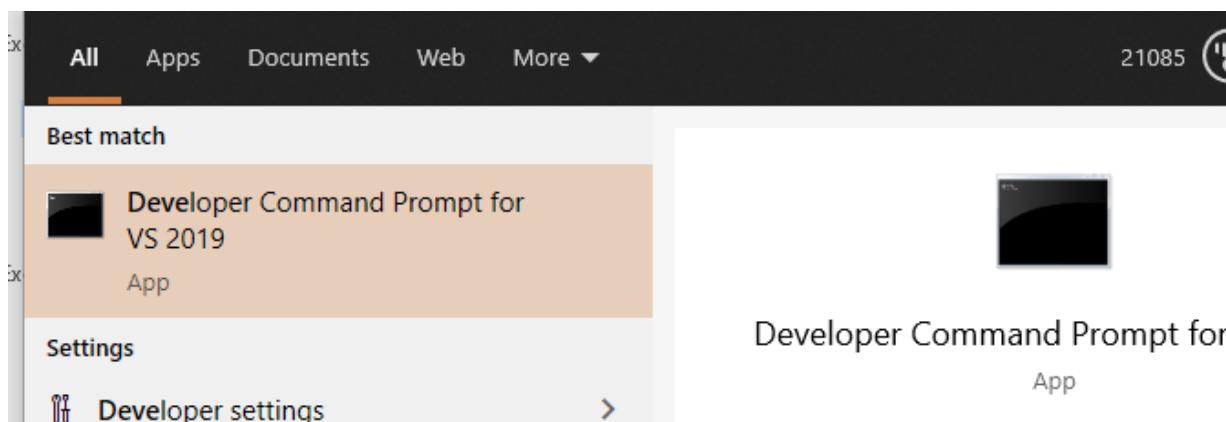
[Download .NET Framework 4.6.2 Developer Pack](#)

- Click **Open file**.

Follow the steps in setup wizard to complete installing the **Developer Pack**.

24.2 Task #2: Setup Components Project

1. Start the developer command prompt tool
 - Click **Start** and search for **developer**.
 - Click to start the **developer command prompt**.



2. Create a new folder in your Documents folder and work in that directory
 - Run the command mentioned below to change directory. Replace **[Computer User Name]** with your OS user name.
`cd C:\Users\[Computer User Name]\Documents`
 - Run the command mentioned below to create a new folder with name **pcfTimelineControl**.
`mkdir pcfTimelineControl`
 - Run the command mentioned below to switch to the **pcfTimelineControl** folder.
`cd pcfTimelineControl`

```
Developer Command Prompt for VS 2019
*****
** Visual Studio 2019 Developer Command Prompt v16.4.3
** Copyright (c) 2019 Microsoft Corporation
*****

C:\Program Files (x86)\Microsoft Visual Studio\2019\Professional>cd C:\Users\Hassan\Documents
C:\Users\Hassan\Documents>mkdir pcfTimelineControl
C:\Users\Hassan\Documents>cd pcfTimelineControl
C:\Users\Hassan\Documents\pcfTimelineControl>
```

3. Create a new folder in the **pcfTimelineControl** folder, name it **src**, and work in that directory

- Create a new folder with the name **src**, by running the command below.

```
mkdir src
```

- Run the command below to switch to the **src** folder you just created.

```
cd src
```

- Clear the screen by running the command below.

```
cls
```

4. Install the latest Power Apps CLI, create a solution project with the **name timelinecontrol**, **namespace contoso**, and **template dataset**.

- Install latest **Power Apps CLI** version. Use: <https://aka.ms/PowerAppsCLI>

Note: if you just installed the tools, you already have the latest, however, you can run this command anytime to ensure you are always up to date.

```
pac install latest
```

- Initialize the component. This command will create a set of files that will implement a dataset component. You will customize these files as per your specific component as we continue.

```
pac pcf init --name timelinecontrol --namespace contoso --template dataset
```

- Install dependencies by running **npm install** command in the Terminal

```
npm install
```

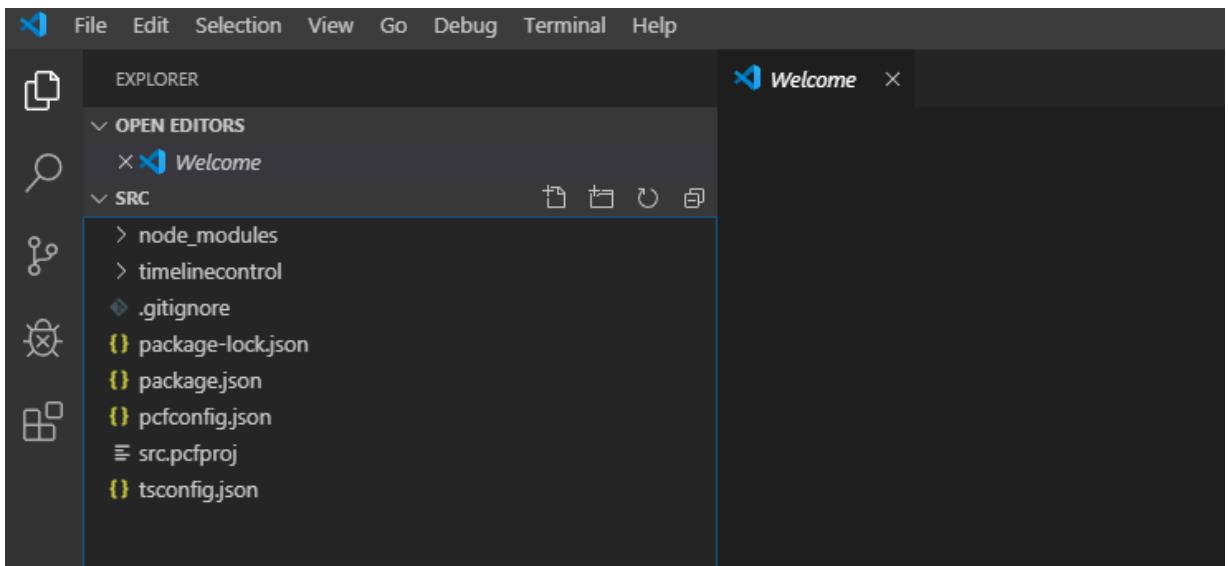
- Wait for the dependency installation to complete.

5. Open the **src** folder in Visual Studio Code and review the generated resources.

- Open the **src** folder in **Visual Studio Code**. For this to work, make sure that the Visual Studio Code is added to Path in Environment Variables.

```
code .
```

- **Visual Studio Code** should start, and it should open the **src** folder.



- Expand the **timelinecontrol** folder.
- Open the **ControlManifest.Input** xml file and examine it.

```

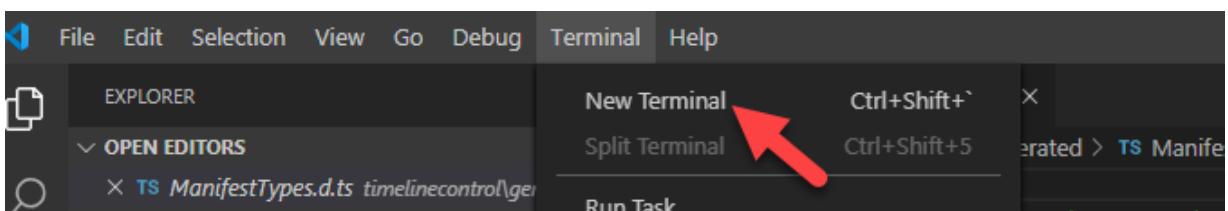
<ControlManifest.Input>
  <manifest>
    <control namespace="contoso" constructor="timelinecontrol" version="0.0.1" display-name-key="TimelineControl">
      <!-- dataset node represents a set of entity records on CDS; allow more than one dataset -->
      <data-set name="sampleDataSet" display-name-key="Dataset_Display_Key">
        <!-- 'property-set' node represents a unique, configurable property that each record is associated with -->
        <property-set name="samplePropertySet" display-name-key="Property_Display_Key" description="A sample property set for the timeline control." />
      </data-set>
      <resources>
        <code path="index.ts" order="1"/>
        <!-- UNCOMMENT TO ADD MORE RESOURCES -->
        <css path="css/timelinecontrol.css" order="1" />
        <resx path="strings/timelinecontrol.1033.resx" version="1.0.0" />
      </resources>
      <!-- UNCOMMENT TO ENABLE THE SPECIFIED API -->
      <feature-usage>
        <uses-feature name="Device.captureAudio" required="true" />
        <uses-feature name="Device.captureImage" required="true" />
        <uses-feature name="Device.captureVideo" required="true" />
        <uses-feature name="Device.getBarcodeValue" required="true" />
        <uses-feature name="Device.getCurrentPosition" required="true" />
        <uses-feature name="Device.pickFile" required="true" />
        <uses-feature name="Utility" required="true" />
        <uses-feature name="WebAPI" required="true" />
      </feature-usage>
    </control>
  </manifest>
</ControlManifest.Input>

```

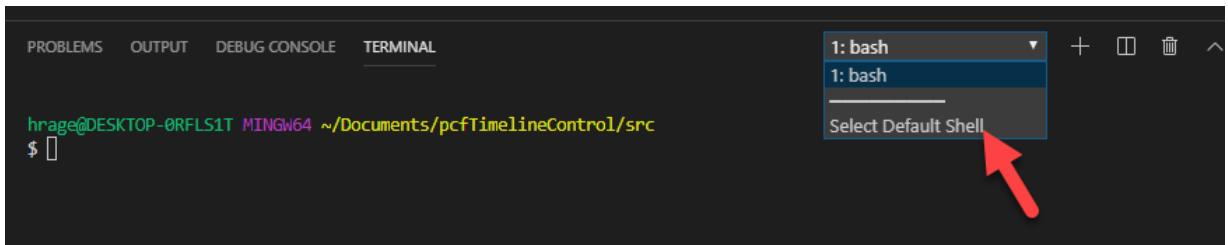
- Open the **Index.ts** file and examine it.
- Expand the **generated** folder.
- Open the **ManifestTypes** file and examine it.

6. Open CLI in visual studio code.

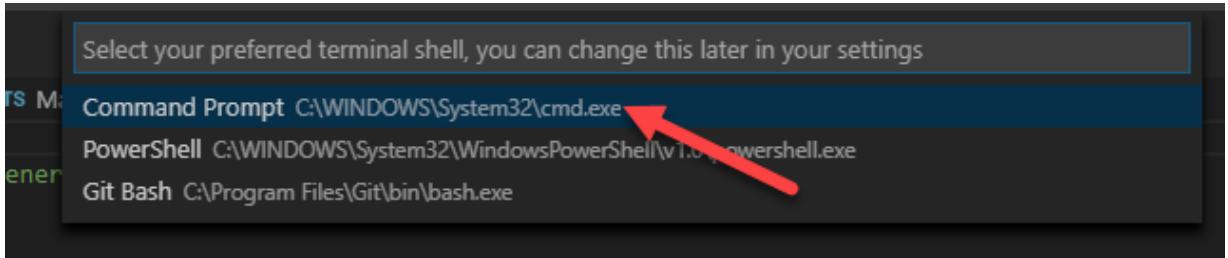
- Click **Terminal** and select **New Terminal**. If Terminal is not visible in the Top menu, you can open it by selecting View -> Integrated Terminal.



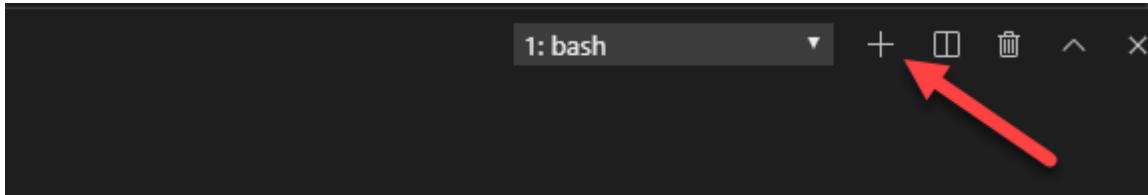
- If cmd isn't your Default Shell open, click on the arrow and click **Select Default Shell**.



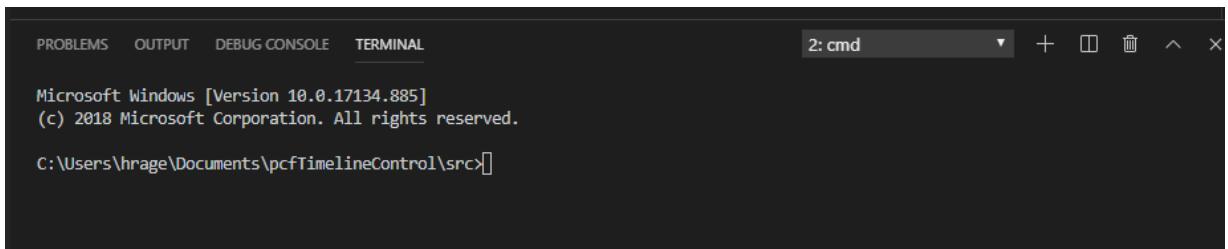
- Select **Command Prompt**.



- Click **New Terminal**.



- The **cmd** terminal should now open.

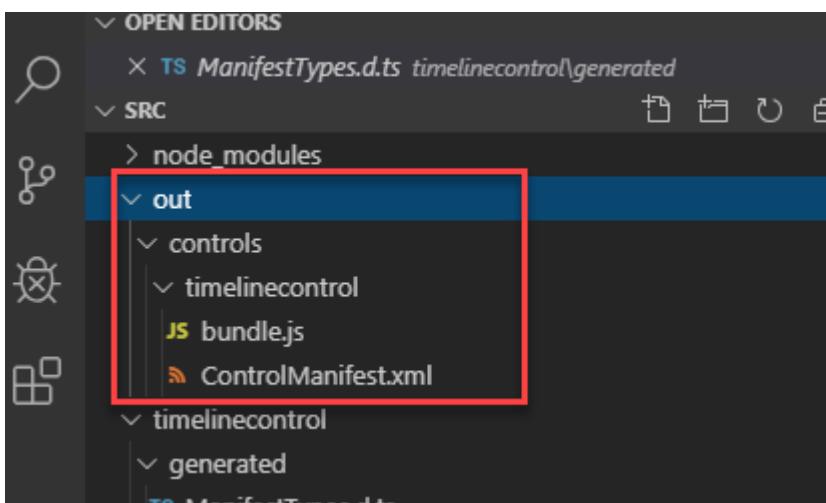


7. Run the Build command and review the out folder.

- Run **npm build** in the terminal

```
npm run build
```

- You should now be able to see the out folder. Expand the out folder and review its content.



8. Run the Start command to start the test harness.

- Run **npm start** in the terminal

```
npm start
```

- This should open the Test Environment in a browser window.

PowerApps component framework Test Environment

timelinecontrol

✓ **Context Inputs**

Form Factor
Web

Component Container Width	Component Container Height
200	200

✓ **Data Inputs**

- The **Component** container size should change, if you provide **Width** and **Height**.

PowerApps component framework Test Environment

timelinecontrol

✓ **Context Inputs**

Form Factor
Web

Component Container Width	Component Container Height
200	200

✓ **Data Inputs**

9. Stop the test harness

- Close the **Test Environment** browser window or tab.
- Go back to **Visual Studio Code**.
- Click on the **Terminal** and press the **[CONTROL]** key and **c**.
- Type **y** and **[ENTER]**.

```
Serving "C:\Users\hrage\Documents\pcfTimelineControl\src\out\controls\timelinecontrol"
Ready for changes
Terminate batch job (Y/N)? y
```

10. Create a new solution folder in the parent of the **src** folder **pcfTimelineControl** and switch to it.

- Change directory to the **pcfTimelineControl** folder.

```
cd ..
```

- You should now be in the **pcfTimelineControl** directory.

```
Terminate batch job (Y/N)? y
```

```
C:\Users\hrage\Documents\pcfTimelineControl\src>cd ..
```

```
C:\Users\hrage\Documents\pcfTimelineControl>[]
```

- Create a new folder with the name **solution**.

```
mkdir solution
```

- Switch to the solution directory.

```
cd solution
```

- You should now be in the solution directory.

```
C:\Users\hrage\Documents\pcfTimelineControl>cd solution
```

```
C:\Users\hrage\Documents\pcfTimelineControl\solution>[]
```

11. Create solution project and add reference of the **src** folder where the component is located to the solution. This configuration will be used when you are done with your development and ready to publish your component for others to use.

- Create solution project with name and prefix contoso.

```
pac solution init --publisher-name contoso --publisher-prefix contoso
```

- Add component location to the solution. This creates a reference to include your component when the solution build command is run.

```
pac solution add-reference --path ..\src
```

- The project reference should be added successfully.

```
C:\Users\hrage\Documents\pcfTimelineControl\solution>pac solution add-reference --path ..\src
Project reference successfully added to CDS solution project.
```

```
C:\Users\hrage\Documents\pcfTimelineControl\solution>[]
```

12. Build the solution

- Make sure you are still in the solution folder.
- Build the project by running the command below.

```
msbuild /t:restore
```

- The build should succeed.

```
Build succeeded.
```

```
0 Warning(s)
```

```
0 Error(s)
```

```
Time Elapsed 00:00:04.52
```

```
C:\Users\hrage\Documents\pcfTimelineControl\solution>[]
```

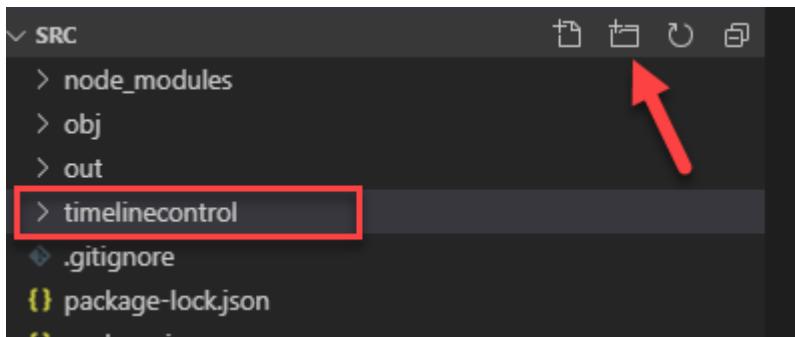
24.3 Task #3: Build the Basic Timeline

1. Change directory to the **src** folder
 - Change directory to the **src** folder.

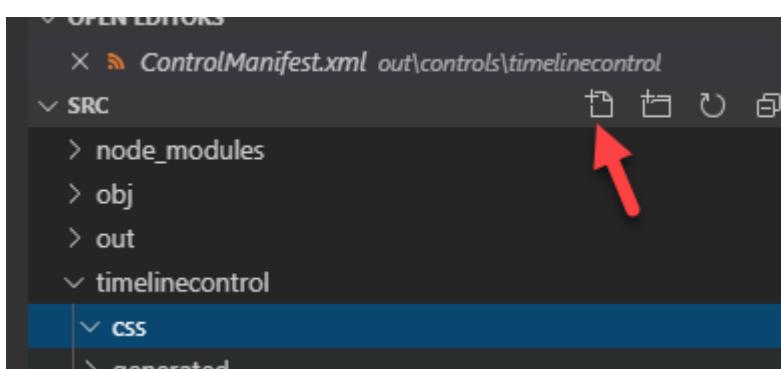
```
cd ..\src
```

2. Create **css** folder in the **timelinecontrol** folder and create ****timelinecontrol.css** file in the **css** folder

- Select the **timelinecontrol** folder and click **New Folder**.



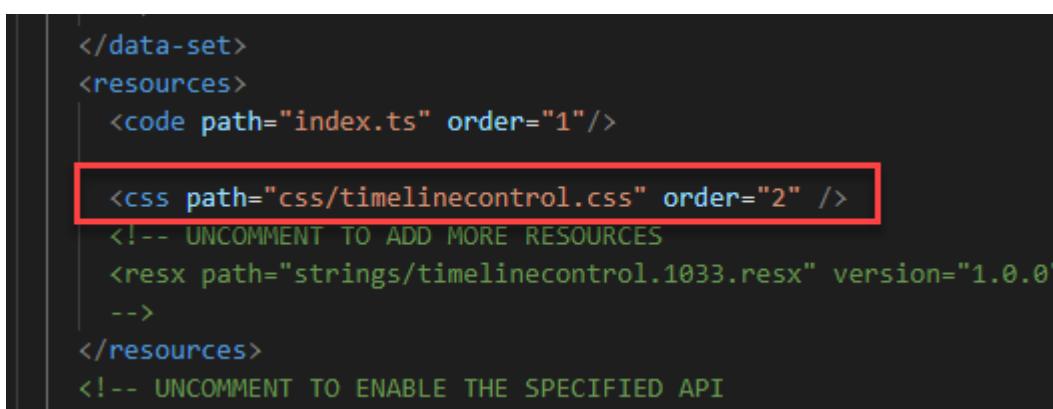
- Name the folder **css**.
- Select the **css** folder you created and click **New File**.



- Name the new file **timelinecontrol.css**.

3. Add the **css** as resource

- Open the **ControlManifest.Input.xml** file.
- Locate the **resources** sub element and uncomment the **css** tag, change the **Order** to **2**.



4. Change the data-set name to **timelineDataSet**.

- Locate **data-set** tag and change the name property to **timelineDataSet**.

```

<manifest>
  <control namespace="contoso" constructor="timelinecontrol" version="0.0.1" >
    <!-- dataset node represents a set of entity records on CDS; allow more than one dataset node per timeline -->
    <data-set name="timelineDataSet" display-name-key="Dataset_Display_Key">
      <!-- 'property-set' node represents a unique, configurable property that can be added to the timeline -->
      <!-- UNCOMMENT TO ADD PROPERTY-SET NODE -->
      <property-set name="samplePropertySet" display-name-key="Property_Display_Key">
        <!-- UNCOMMENT TO ADD PROPERTY-SET NODE -->
    </data-set>
  </control>
</manifest>

```

5. Install vis-timeline css npm package

- Go to the **Terminal** and make sure you are in the **src** directory.
- Run the command mentioned below and wait for the packages to be added.

```
npm install vis-timeline
```

```
+ vis-timeline@5.1.0
added 9 packages from 139 contributors and audited 10285 packages in 8.131s
found 0 vulnerabilities
```

```
C:\Users\hrage\Documents\pcfTimelineControl\src>
```

- Run the command mentioned below and wait for the packages to be added.

```
npm install moment
```

- Run the command mentioned below and wait for the packages to be added.

```
npm install vis-data
```

6. Add the vis-timeline css as a resource

- Go back to the **ControlManifest.Input.xml** file.
- Add the vis-timeline css inside the resources tag.

```

<css path=".\\node_modules\\vis-timeline\\dist\\vis-timeline-graph2d.min.css" order="1" />
<!-- UNCOMMENT TO ADD MORE RESOURCES -->
<resources>
  <code path="index.ts" order="1"/>
  <css path=".\\node_modules\\vis-timeline\\dist\\vis-timeline-graph2d.min.css" order="1" />
  <css path="css/timelinecontrol.css" order= 2 />
  <!-- UNCOMMENT TO ADD MORE RESOURCES -->
  <resx path="strings/timelinecontrol.1033.resx" version="1.0.0" />
  <!-- UNCOMMENT TO ENABLE THE SPECIFIED API -->
  <feature_usage>
    <!-- UNCOMMENT TO ADD MORE FEATURES -->
  </feature_usage>
</resources>

```

7. Add timeline element and visual properties to the Index file

- Open the **Index.ts** file.
- Add the properties below, inside the **export** class timelinecontrol function.

```
private _timelineElm: HTMLDivElement;
private _timelineVis : any;
```

```

5   export class timelinecontrol implements ComponentFramework.StandardControl<IInputs, IOOutputs> {
6
7     private _timelineElm: HTMLDivElement;
8     private _timelineVis : any;
9
10    /**
11     * Empty constructor.
12     */
13    constructor()
14  {

```

- Add the below constant after the import lines on the top.

```
const vis = require('vis-timeline');

import {IInputs, IOOutputs} from "./generated/ManifestTypes";
import DataSetInterfaces = ComponentFramework.PropertyHelper.DataSetApi;
const vis = require('vis-timeline');

type DataSet = ComponentFramework.PropertyType.TypesDataSet;
```

8. Build the timeline element as div and add it to container element as a child.

- Locate the **init** method.
- Add the script mentioned below to the **init** function.

```
this._timelineElm = document.createElement("div");

container.appendChild(this._timelineElm);

/* @param container If a control is marked control-type='standard', it will receive a
 */
public init(context: ComponentFramework.Context<IInputs>, notifyOutputChanged: () => void) {
    // Add control initialization code
    this._timelineElm = document.createElement("div");
    container.appendChild(this._timelineElm);
}
```

9. Create a function that will render the timeline

- Add the function below.

```
private renderTimeline(): void {
    // Create a DataSet (allows two way data-binding)
    var items = [
        { id: 1, content: 'item 1', start: '2020-08-20' },
        { id: 2, content: 'item 2', start: '2020-08-14' },
        { id: 3, content: 'item 3', start: '2020-08-18' },
        { id: 4, content: 'item 4', start: '2020-08-16', end: '2020-08-19' },
        { id: 5, content: 'item 5', start: '2020-08-25' },
        { id: 6, content: 'item 6', start: '2020-08-27', type: 'point' }
    ];
    // Configuration for the Timeline
    var options = {};
    // Create a Timeline
    var timeline = new vis.Timeline(this._timelineElm, items, options);
}
```

```

private renderTimeline(): void {
    // Create a DataSet (allows two way data-binding)
    var items = [
        { id: 1, content: 'item 1', start: '2020-08-20' },
        { id: 2, content: 'item 2', start: '2020-08-14' },
        { id: 3, content: 'item 3', start: '2020-08-18' },
        { id: 4, content: 'item 4', start: '2020-08-16', end: '2020-08-19' },
        { id: 5, content: 'item 5', start: '2020-08-25' },
        { id: 6, content: 'item 6', start: '2020-08-27', type: 'point' }
    ];
    // Configuration for the Timeline
    var options = {};
    // Create a Timeline
    var timeline = new vis.Timeline(this._timelineElm, items, options);
}

```

10. Call the `renderTimeline` function from the `updateView` function.

- Locate the `updateView` function.
- Add the script mentioned below inside the `updateView` function.

```

this.renderTimeline();
/* @param context - the entire property bag available to control via context object */
public updateView(context: ComponentFramework.Context<IInputs>): void {
    // Add code to update control view
    this.renderTimeline();
}

private renderTimeline(): void {

```

- Click **File** and **Save All**.

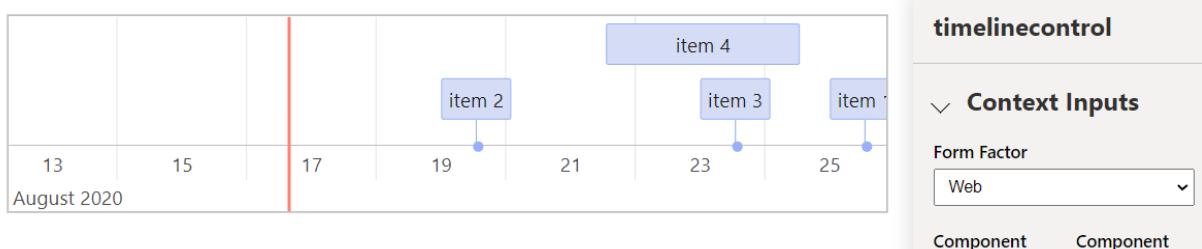
11. Build and start

- Go to the **Terminal** and make sure you are still in the `src` directory.
- Run the build command.

```
npm run build
```

- The build should succeed.
- Run the start watch command. This command will keep the test environment running and auto update when you change the component.

```
npm start watch
```



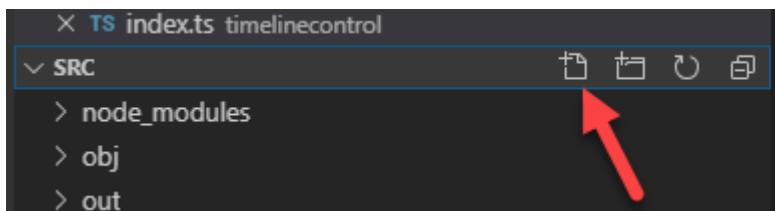
Do not close the test environment.

24.4 Task #4: Tailor for Inspection Data

In this task, you will switch from using the hard-coded array of data to using a file loaded into the test harness.

1. Create test data csv file

- Select the **src** folder. And click **New File**.



- Name the new file **testdata.csv**
- Add the below mentioned data inside the **testdata.csv** file and Save it.

```
contoso_permitid,contoso_name,contoso_scheduleddate,statuscode
123,Electrical:Rough Inspection:Passed,8/1/2020,Passed
124,Electrical:Rough Inspection:Passed,8/5/2020,Passed
125,Plumbing:Rough Inspection:Failed,8/8/2020,Failed
126,Plumbing:Rough Inspection:Passed,8/10/2020,Passed
```

A screenshot of a code editor showing a file named "testdata.csv". The file contains the following CSV data:

```
contoso_permitid,contoso_name,contoso_scheduleddate,statuscode
1 123,Electrical:Rough Inspection:Passed,8/1/2020,Passed
2 124,Electrical:Rough Inspection:Passed,8/5/2020,Passed
3 125,Plumbing:Rough Inspection:Failed,8/8/2020,Failed
4 126,Plumbing:Rough Inspection:Passed,8/10/2020,Passed
5
```

The first row is a header, and the subsequent rows are data entries. The first column is labeled "contoso_permitid" and has values 1, 2, 3, 4, and 5 respectively. The second column is labeled "contoso_name" and has values "123,Electrical:Rough Inspection:Passed,8/1/2020,Passed", "124,Electrical:Rough Inspection:Passed,8/5/2020,Passed", "125,Plumbing:Rough Inspection:Failed,8/8/2020,Failed", "126,Plumbing:Rough Inspection:Passed,8/10/2020,Passed", and an empty line. The third column is labeled "contoso_scheduleddate" and has values "statuscode" and "123,Electrical:Rough Inspection:Passed,8/1/2020,Passed" respectively. The fourth column is labeled "statuscode" and has values "123,Electrical:Rough Inspection:Passed,8/1/2020,Passed", "124,Electrical:Rough Inspection:Passed,8/5/2020,Passed", "125,Plumbing:Rough Inspection:Failed,8/8/2020,Failed", "126,Plumbing:Rough Inspection:Passed,8/10/2020,Passed", and an empty line.

2. Create Timeline Data class

- Open the **index.ts** file.
- Paste the code below after the **type DataSet** line.

```
class TimelineData {  
    id: string;  
    content: string;  
    start: string;  
    className: string;  
  
    constructor(id: string, content: string, start: string, className: string) {  
        this.id = id;  
        this.content = content;  
        this.start = start;  
        this.className = className;  
    }  
}
```

```

type DataSet = ComponentFramework.PropertyTypes.DataSet;

class TimelineData {
    id: string;
    content: string;
    start: string;
    className: string;

    constructor(id: string, content: string, start: string, className: string) {
        this.id = id;
        this.content = content;
        this.start = start;
        this.className = className;
    }
}

export class timelinecontrol implements ComponentFramework.StandardControl<IInputs, IOutputs> {

```

- Add the timeline data array property inside the **export** class timelinecontrol function and below the **_timelineElm** definition.

```

    private _timelineData : TimelineData[] = [];

20   export class timelinecontrol implements ComponentFramework.StandardControl<IInputs, IOutputs> {
21
22     private _timelineElm: HTMLDivElement;
23     private _timelineData : TimelineData[] = [];
24     private _timelineVis: any;
25
26     /**
27      * Empty constructor.
28

```

3. Add a method that will create the timeline data.

- Add the method mentioned below after the **render** method.

```

    private createTimelineData(gridParam: DataSet) {
        this._timelineData = [];
        if (gridParam.sortedRecordIds.length > 0) {
            for (let currentRecordId of gridParam.sortedRecordIds) {

                console.log('record: ' + gridParam.records[currentRecordId].getRecordId());

                var permitName = gridParam.records[currentRecordId].getFormattedValue('contoso_name');
                var permitDate = gridParam.records[currentRecordId].getFormattedValue('contoso_startdate');
                var permitStatus = gridParam.records[currentRecordId].getFormattedValue('statuscode');
                var permitColor = "green";
                if (permitStatus == "Failed")
                    permitColor = "red";
                else if (permitStatus == "Canceled")
                    permitColor = "yellow";

                console.log('name:' + permitName + ' date:' + permitDate);

                if (permitName != null)
                    this._timelineData.push(new TimelineData(currentRecordId, permitName, permitDate));
            }
        }
    else {
        //handle no data
    }
}

```

```

    }

private createTimelineData(gridParam: DataSet) {
    this._timelineData = [];
    if (gridParam.sortedRecordIds.length > 0) {
        for (let currentRecordId of gridParam.sortedRecordIds) {

            console.log('record: ' + gridParam.records[currentRecordId].getRecordId());

            var permitName = gridParam.records[currentRecordId].getFormattedValue('contoso_name')
            var permitDate = gridParam.records[currentRecordId].getFormattedValue('contoso_scheduleddate')
            var permitStatus = gridParam.records[currentRecordId].getFormattedValue('statuscode')
            var permitColor = "green";
            if (permitStatus == "Failed")
                permitColor = "red";
            else if (permitStatus == "Canceled")
                permitColor = "yellow";

            console.log('name:' + permitName + ' date:' + permitDate);

            if (permitName != null)
                this._timelineData.push(new TimelineData(currentRecordId, permitName, permitDate, permitColor));
        }
    } else {
        //handle no data
    }
}

```

4. Call the `createTimelineData` method from the `updateView` method.

- Go to the `updateView` method.
- Replace the code inside the `updateView` method with the code below.

```

if (!context.parameters.timelineDataSet.loading) {
    // Get sorted columns on View
    this.createTimelineData(context.parameters.timelineDataSet);
    this.renderTimeline();
}

public updateView(context: ComponentFramework.Context<IInputs>): void {
    // Add code to update control view
    if (!context.parameters.timelineDataSet.loading) {
        // Get sorted columns on View
        this.createTimelineData(context.parameters.timelineDataSet);
        this.renderTimeline();
    }
}

```

5. Replace the hardcoded items with the csv data.

- Locate the `renderTimeline` method.
- Replace the hardcoded `items` with code below.

```
var items = this._timelineData;
```

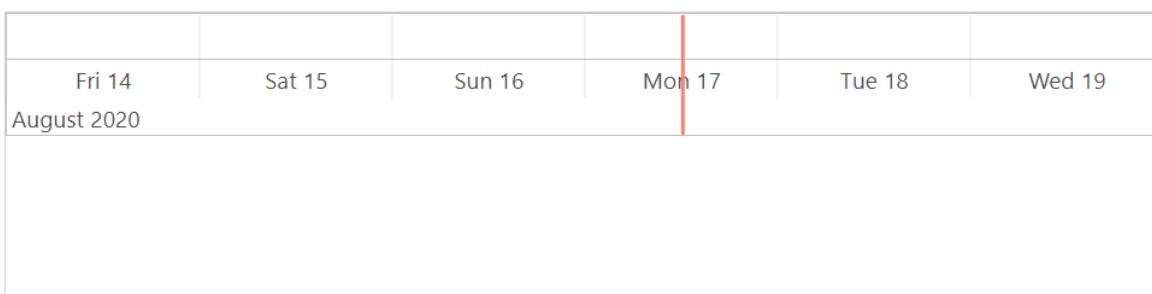
```

private renderTimeline(): void {
    // Create a DataSet (allows two way data-binding)
    var items = this._timelineData;
    // Configuration for the Timeline
    var options = {};
    // Create a Timeline
    var timeline = new vis.Timeline(this._timelineElm, items, options);
}

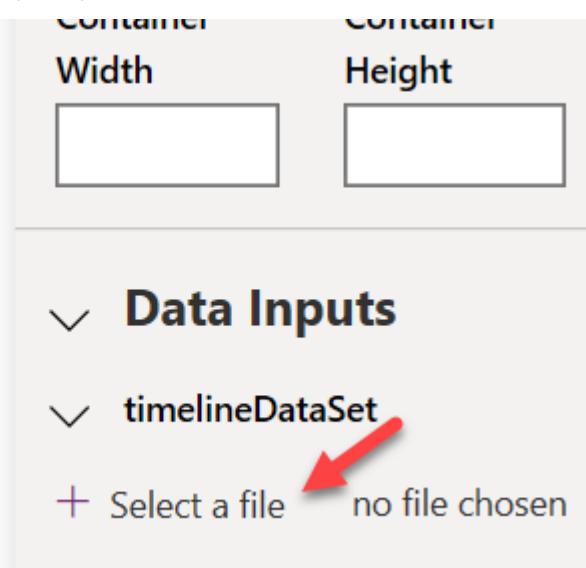
```

6. Make sure the test environment shows your changes and test the timeline control with the test data.

- Click **File** and then **Save All**.
- The test harness should still be running. If it is not running run **npm start watch** command.
- Go to the test environment and make sure it looks like the image below.



- Click **Select a File**.



- Select the **testdata.csv** and click **Open**.

out	8/13/2020 12:21 PM	File folder
timelinecontrol	8/17/2020 11:50 AM	File folder
testdata.csv	8/17/2020 11:58 AM	Microsoft Excel C... 1 KB



- Click **Apply**.

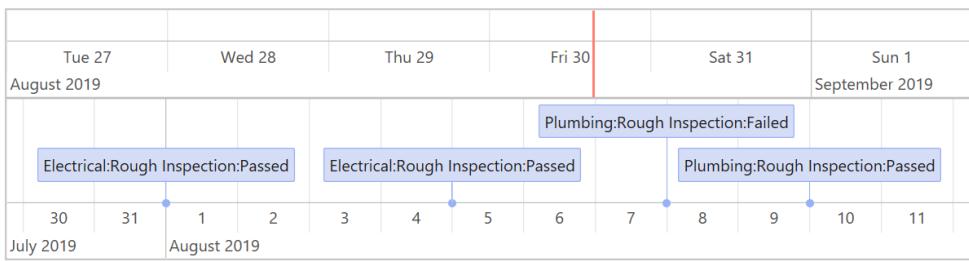
statuscode

Type

SingleLine.Text

Apply

- The timeline control should now show the test data.



timelinecontrol

Context Inputs

Form Factor: Web

Component Container Width: []

Component Container Height: []

Do not close the test environment.

24.5 Task #5: Change Color for Items

In this task, you will use the **css** resource you configured to change the color of the items on the timeline.

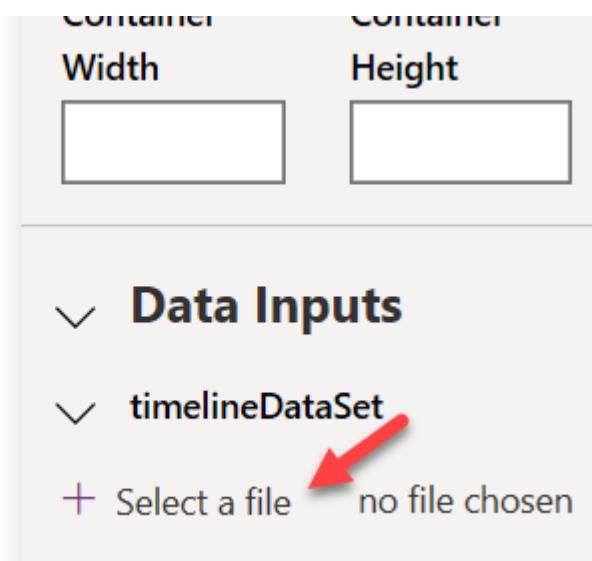
- Add red and green styles to the **timelinecontrol.css** file
 - Go back to **Visual Studio Code**.
 - Expand the **css** folder and open the **timelinecontrol.css**
 - Add the style below to the **timelinecontrol.css** file and save your changes.

```
.red{
    background:red;
    color:white;
}
.green{
    background:green;
    color:white;
}
.yellow{
    background:yellow;
    color:black;
}
```

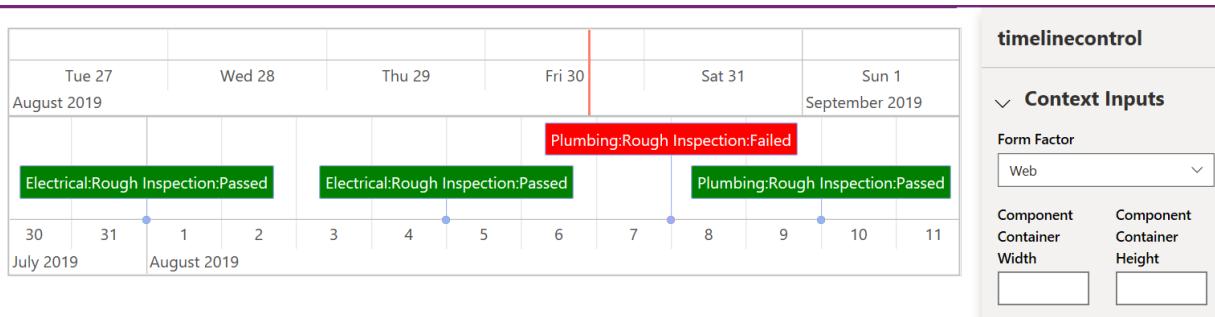
```
# timelinecontrol.css ×  TS index.ts
src > timelinecontrol > css > # timelinecontrol.css > ...
1  .red{
2    background:red;
3    color:white;
4  }
5  .green{
6    background:green;
7    color:white;
8  }
9  .yellow{
10   background:yellow;
11   color:black;
12 }
```

- Check the test environment, load the test data and make sure it shows your changes

- Go to the **Test Environment**.
- Click **Select a File**.



- Select the **testdata.csv** and click **Open**.
- Click **Apply**.
- The timeline control should now show the test data.



- Close the test environment browser window or tab.

- Stop the test

- Go back to **Visual Studio Code**.

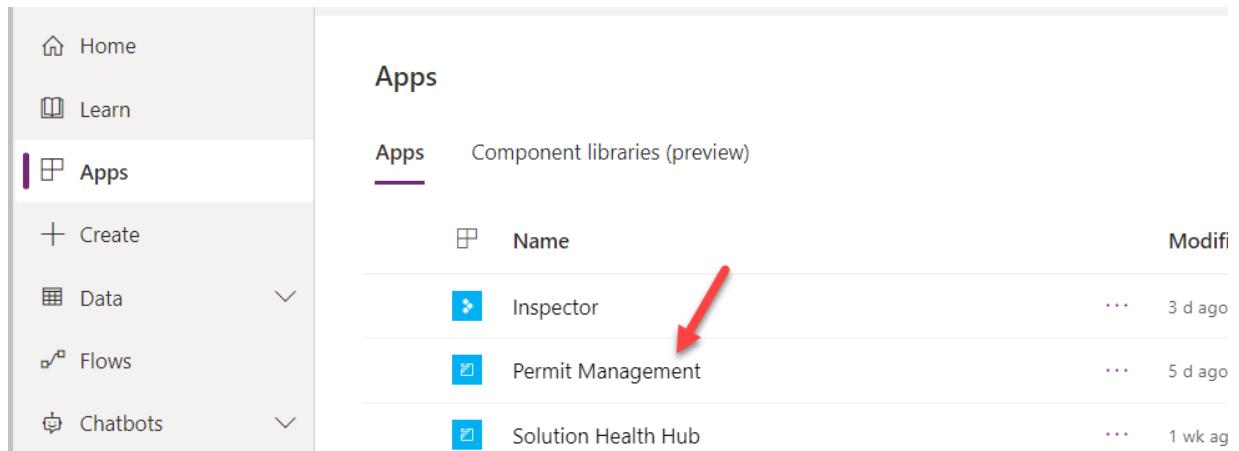
- Click on the **Terminal** and press the **[CONTROL]** key and **c**.
- Type **y** and **[ENTER]**.

25 Exercise #2: Publish to Microsoft Dataverse

Objective: In this exercise, you will publish the timeline control to your Microsoft Dataverse and add it to the Permit main form.

25.1 Task #1: Setup and Publish

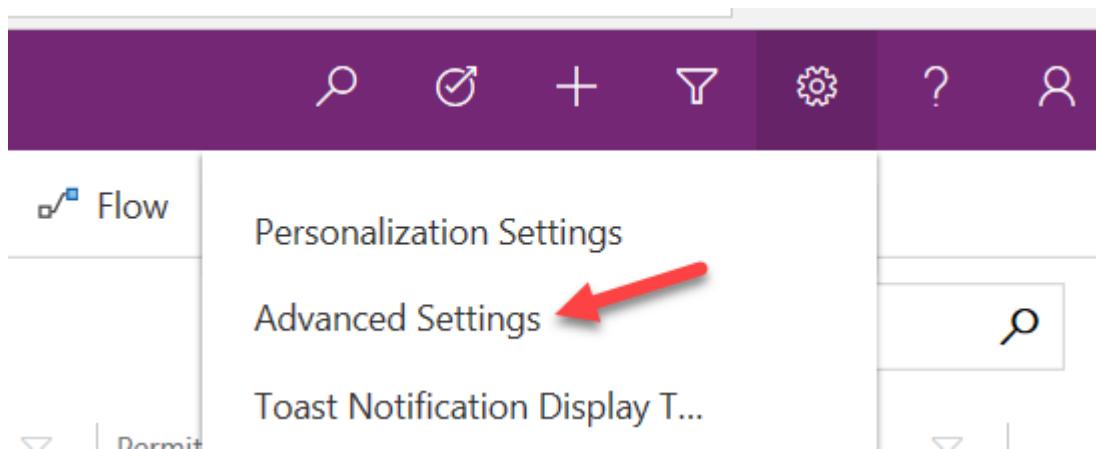
1. Get your Microsoft Dataverse org URL
 - Navigate to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
 - Select **Apps** and open the Permit Management application.



The screenshot shows the Microsoft Dataverse Apps screen. On the left is a navigation sidebar with options: Home, Learn, Apps (which is selected and highlighted in purple), Create, Data, Flows, and Chatbots. The main area is titled "Apps" and shows a list of apps. The columns are "Name", "Modified", and "...". The apps listed are Inspector (modified 3 d ago), Permit Management (modified 5 d ago), and Solution Health Hub (modified 1 wk ago). A red arrow points to the "Permit Management" app.

Name	Modified	...
Inspector	3 d ago	...
Permit Management	5 d ago	...
Solution Health Hub	1 wk ago	...

- Click Settings and select Advanced Settings.



The screenshot shows the settings screen for the Permit Management app. At the top is a purple header bar with icons for search, refresh, plus, filter, gear, help, and user. Below the header is a sidebar with "Flow" and "Permit" sections. The main content area has three items: "Personalization Settings", "Advanced Settings" (which has a red arrow pointing to it), and "Toast Notification Display T...".

- Navigate to **Settings | Customizations**.

Dynamics 365 Settings

Business

- Business Management...
- Templates

Customization

- Customizations
- Solutions

System

- Administration
- Security

- Click **Developer Resources**.
- Copy your organization **URL**.

Organization Service

SOAP Service providing access to this instance of Dynamics 365. For more information see [Use the IOrganizationService web service to read and write data or metadata.](#)

Endpoint Address <https://orga5253b09.api.crm.dynamics.com/XRMServices/2011/Organization.s>

2. Authenticate

- Go back to **Visual Studio Code**.
 - Make sure you are still in the **src** directory.
 - Run the command below. Replace <orgurl> with **URL** you copied.
- ```
pac auth create --url <orgurl>
```
- Sign in with your **admin** username.

## 3. Import the solution into your org and publish

- Run the command below and wait for the publishing to complete. The push command uploads your component to the configured environment. This can be used over and over during development to quickly see your component in the live form.

```
pac pcf push --publisher-prefix contoso
```

```
Waiting on completion of import job 'edded60d9-558a-43f9-94cc-e0369905ea91': 3/7, 42%
Processing of import job ended at 8/30/2019 4:06 PM after 100% completion: 4/7, 57%
Import job reported a status of 'SUCCESS': 5/7, 71%.
Unmanaged solution import has completed, starting to publish customizations: 6/7, 8%.
Customizations have been published: 7/7, 100%.
Importing the temporary solution wrapper into the current org: done.

C:\Users\hrage\Documents\pcfTimelineControl\src>[]
```

## 25.2 Task #2: Add Timeline Control to the Permit Form

### 1. Open the Permit Management solution.

- Navigate to **Power Apps maker portal** and make sure you have the **Dev** environment selected.

- Select **Solutions**.
- Click to open the **Permit Management** solution.

The screenshot shows the 'PowerApps Checker' interface. On the left, there's a sidebar with 'AI Builder (preview)' and 'Solutions'. Under 'Solutions', 'Permit Management' is highlighted with a red arrow. The main area lists several solutions: 'PowerApps Checker Base', 'PowerApps Checker Update', 'PowerApps Checker', and 'Permit Management' (with a red arrow pointing to it), followed by 'Common Data Services Default Solution'.

2. Open the Permit Main form and switch to classic

- Locate and click to open the **Permit** Table.

|                     |     |                       |              |  |          |
|---------------------|-----|-----------------------|--------------|--|----------|
| Inspection Type     | ... | contoso_inspectionony | Option Set   |  | -        |
| Inspector           | ... | contoso_inspector_cd  | Canvas App   |  | 1 d ago  |
| Permit              | ... | contoso_permit        | Entity       |  | -        |
| Permit Form Scripts | ... | contoso_PermitFormS   | Web Resource |  | 18 h ago |

- Select the **Forms** tab.
- Click to open the **Main** form.

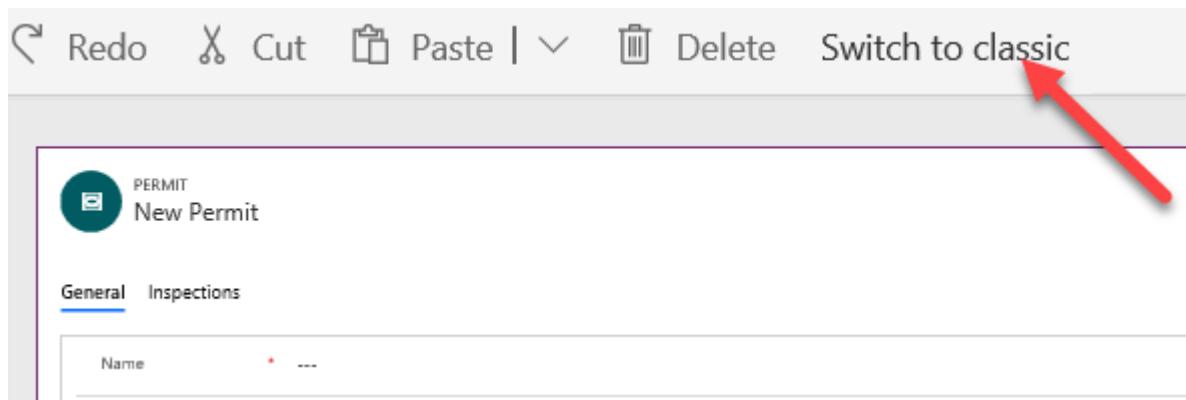
Solutions > Permit Management > Permit

Fields   Relationships   Business rules   Views   **Forms**   Dashboards   Charts   Keys   Data

Model-driven

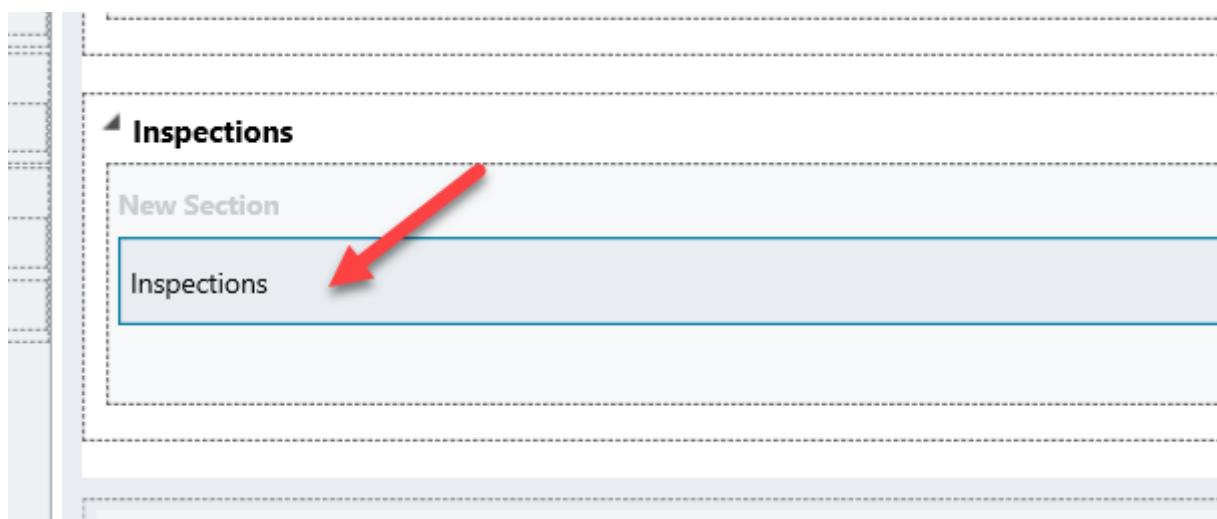
| Name ↑      | Form type ↘ | Type ↘           |
|-------------|-------------|------------------|
| Information | ...         | Main             |
| Information | ...         | Quick View Fo... |
| Information | ...         | Card             |

- Click **Switch to Classic**.

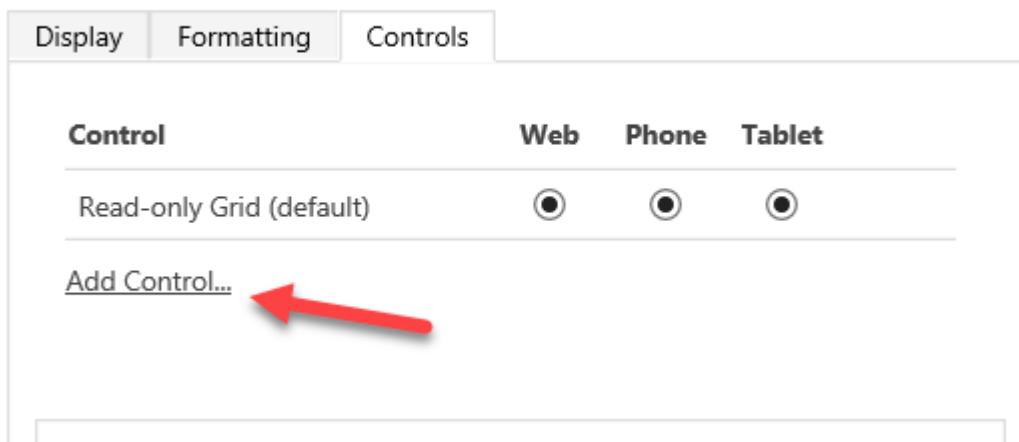


3. Add Timeline control to the form

- Locate the **Inspections** tab.
- Double click on the **Inspections** sub-grid.



- Select the **Controls** tab and click **Add Control**.



- Select **timelinecontrol** and click Add.

## Timeline control

timelinecontrol

### timelinecontrol

Modes:

Types: Grid

timelinecontrol description



Add

- Select **Web** and click **OK**.

| Control                  | Web                              | Phone                            | Tablet                           |
|--------------------------|----------------------------------|----------------------------------|----------------------------------|
| Read-only Grid (default) | <input type="radio"/>            | <input checked="" type="radio"/> | <input checked="" type="radio"/> |
| timelinecontrol          | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            |

[Add Control...](#)

timelinecontrol

| Property            | Value              |
|---------------------|--------------------|
| Dataset_Display_Key | Active Inspections |

OK

Cancel

4. Save and publish

- Click **Save**.
- Click **Publish** and wait for the publishing to complete.
- **DO NOT** close the form editor.

5. Create test data

- Go back to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select **Apps** and click to start the **Permit Management** application

- Click Advanced Find.



Templates Export to Excel ...

Search for records



- Select Inspections and click Results.

**FILE ADVANCED FIND**

Query Saved Views Results New Save Save As Edit Columns Edit Properties Clear Group AND Group OR Details Download XML Query Debug

Show View

Look for: Inspections Use Saved View: [new]

Select

- Click to open the **Framing Inspection**.
- Change the **Status Reason** to **Passed** and click **New**.

+ New Deactivate Delete Refresh Assign Share Email

**INSPECTION**  
Framing Inspection

**General** Related

|               |        |
|---------------|--------|
| Status Reason | Passed |
|---------------|--------|

- Provide a Name, select Inspection Type, select the Test Permit, select Scheduled Date, select Failed for Status Reason, and click **Save and Close**.

Save Save & Close New Flow

New Inspection

Failed  
Status Reason

**General**

|                 |                       |
|-----------------|-----------------------|
| Name            | * Electric Inspection |
| Inspection Type | Initial Inspection    |
| Permit          | Test Permit           |
| Scheduled Date  | * 8/19/2020           |
| Owner           | *  MOD Administrator  |
| Comments        | ---                   |

- Close Advanced Find.

6. Test the control

- Click to open a **Permit** record.

Pinned

Permits

Permits

Inspections

Test Permit

- Select the **Inspections** tab.
- The control should show the two inspections, but the color will not match the status reason values.

The screenshot shows the 'Inspections' tab for a 'Test Permit'. The timeline grid has two entries: 'Framing Inspection' at 00:00 on Fri 30 August and 'Electric Inspections' at 00:00 on Sat 31 August.

### 25.3 Task #3: Debug

- Start Edge DevTools and add breakpoint.

- Press **F12**.
- Search for `createTimelineData = function`
- Locate the `createTimelineData` function and add breakpoint on the `permitColor = "green"` line.

```

69
70 timelinecontrol.prototype.createTimelineData = function (gridParam) {
71 this._timelineData = [];
72
73 if (gridParam.sortedRecordIds.length > 0) {
74 for (var _i = 0, _a = gridParam.sortedRecordIds; _i < _a.length; _i++) {
75 var currentRecordId = _a[_i];
76 console.log('record: ' + gridParam.records[currentRecordId].getRecordId());
77 var permitName = gridParam.records[currentRecordId].getFormattedValue('contoso_name');
78 var permitDate = gridParam.records[currentRecordId].getFormattedValue('contoso_scheduleddate');
79 var permitStatus = gridParam.records[currentRecordId].getFormattedValue('statuscode');
80 var permitColor = "green";
81 if (permitStatus == "Failed") permitColor = "red";

```

- Go back to the Permit Management application and click Refresh.

The screenshot shows the 'Inspections' tab for a 'Test Permit'. The top navigation bar includes buttons for New, Deactivate, Lock Permit, Delete, Refresh (with a red arrow pointing to it), Assign, Share, and Email.

The screenshot shows the 'Inspections' tab for a 'Test Permit'. The top navigation bar includes buttons for General, Inspections (with a red arrow pointing to it), and Related.

- Select the **Inspections** tab again.
- Execution should break.
- Hover over the `permitStatus`, the `permitStatus` is null because **Status Reason** is not included in the **View**.

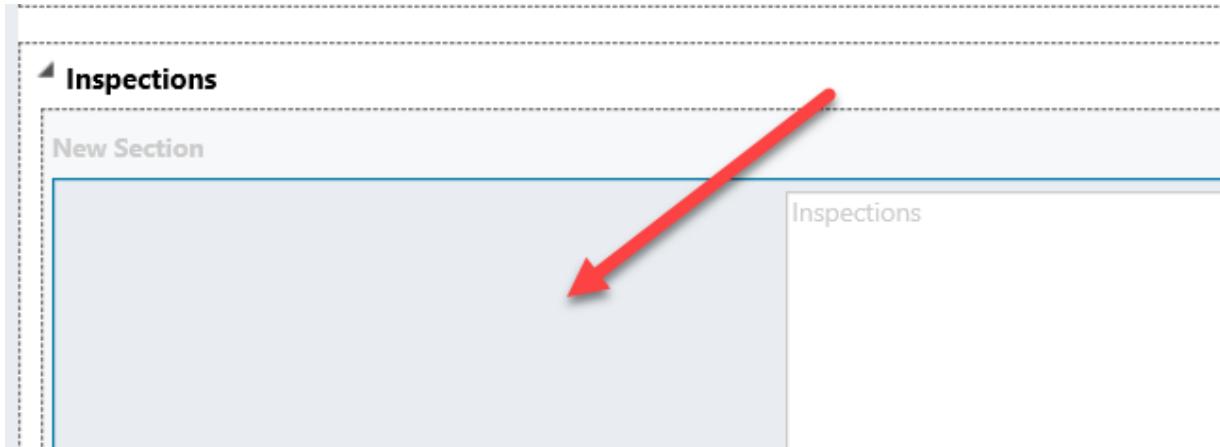
```
76 console.log('record: ' + gridParam.records[currentRecordId].getRecordId());
77 var permitName = gridParam.records[currentRecordId].getFormattedValue('contoso_name');
78 var permitDate = gridParam.records[currentRecordId].getFormattedValue('contoso_schedule');
79 var permitStatus = gridParam.records[currentRecordId].getFormattedValue('statuscode');
80 var permitC
81 if (permits) { permitStatus = null }
```

- Press **F5** to continue.

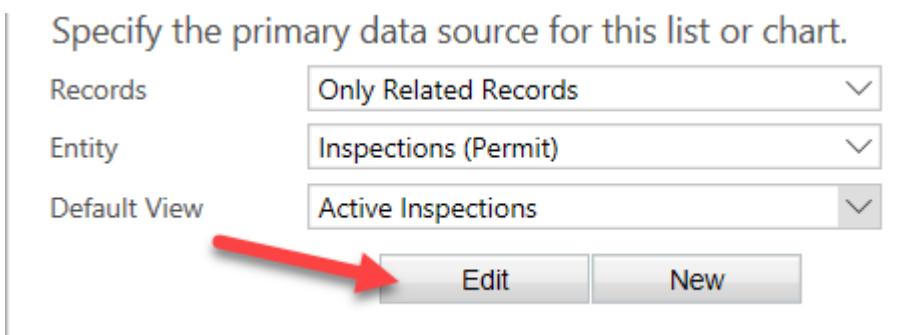
- Close the **DevTools**.

## 2. Add Status Reason to the view.

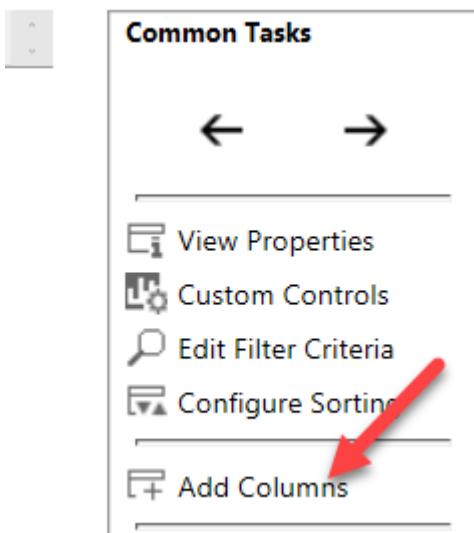
- Go back to the form editor.
- Double click on the Inspections timeline control.



- Click **Edit**.

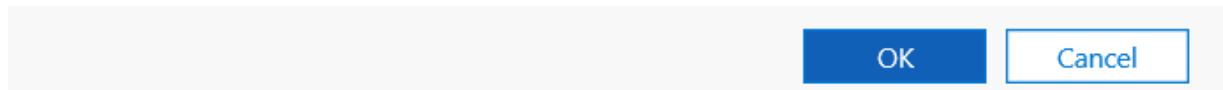


- Click **Add Column**.



- Select **Status Reason** and click **OK**.

|                                                   |                     |               |
|---------------------------------------------------|---------------------|---------------|
| <input type="checkbox"/> Permit                   | contoso_permit      | LOOKUP        |
| <input type="checkbox"/> Record Created On        | overriddencreatedon | Date and Time |
| <input type="checkbox"/> Status                   | statecode           | Status        |
| <input checked="" type="checkbox"/> Status Reason | statuscode          | Status Reason |



- Move the **Status Reason** Column after the **Name** Column.

The screenshot shows the 'View: Active Inspections' configuration screen. A red box highlights the 'Status Reason' column header in the list view. To the right, a 'Common Tasks' ribbon is open, with a red arrow pointing to the left arrow icon, indicating the move action.

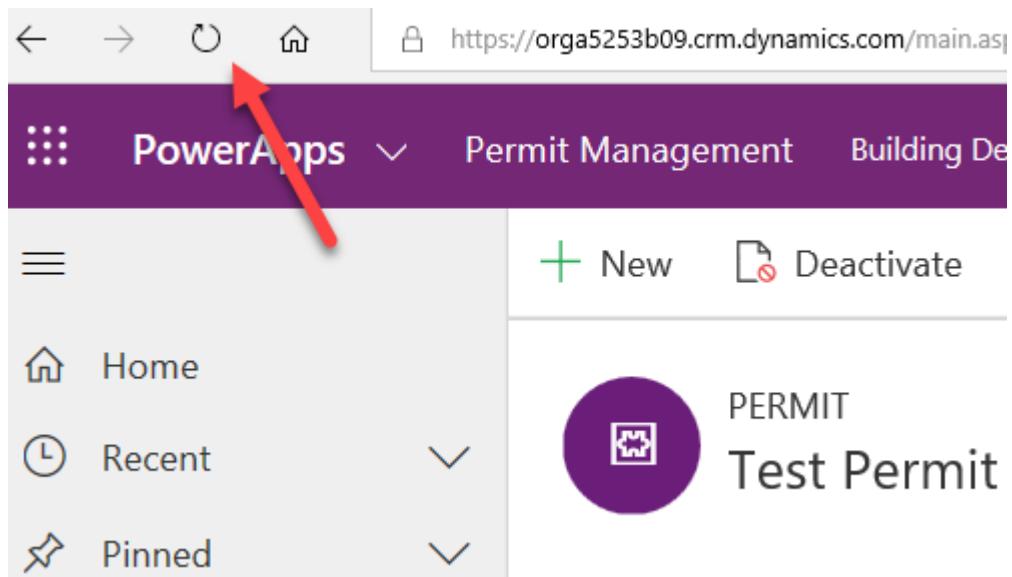
- Click **Save and Close**.
- Click **OK**.

The screenshot shows the 'Active Inspections' view configuration screen. It includes a 'Default View' section with 'Edit' and 'New' buttons, and an 'Additional Options' section with checkboxes for 'Display Search Box' and 'Display Index'. At the bottom are 'OK' and 'Cancel' buttons.

- Click **Save**.
- Click **Publish** and wait for the publishing to complete.
- Close the form editor.

### 3. Test your changes

- Go back to the **Permit Management** application and refresh the browser.



- Select the **Inspections** tab. The timeline control should now show the correct colors.

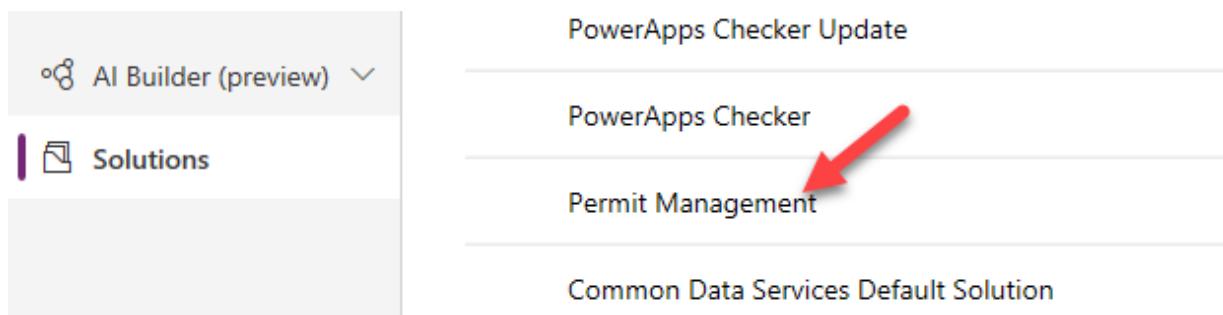
General   **Inspections**   Related



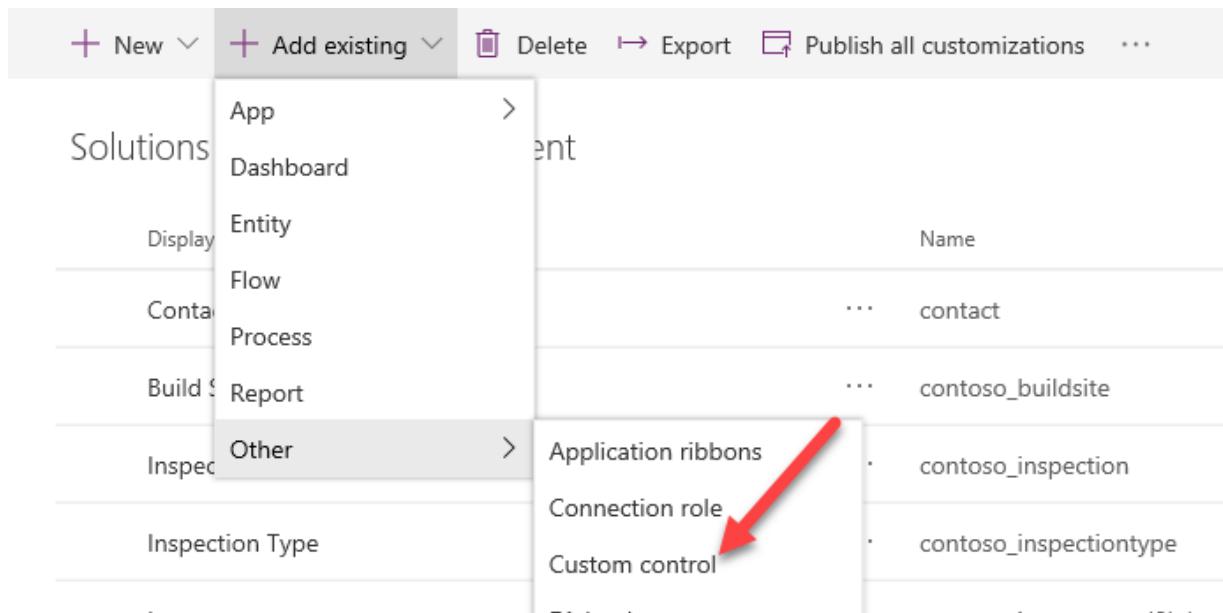
## 25.4 Task #4: Add the Timeline Control to Permit Management Solution

### 1. Add Custom Control to solution

- Navigate to [Power Apps maker portal - screenshot](#) and make sure you are in the **Dev** environment.
- Select **Solutions** and click to open the **Permit Management** solution.



- Click **Add Existing | Other | Custom Control**.



- Search for Timeline, select **contoso\_contoso.timelinecontrol** and click **Add**.

#### Add existing custom controls

Select custom controls from other solutions or custom controls that aren't in solutions yet. Adding custom controls that aren't already in solutions will also add them to Common Data Service.

| 1 custom control selected |                                 | Custom control | Timeline                        |
|---------------------------|---------------------------------|----------------|---------------------------------|
| Display name              | contoso_contoso.timelinecontrol | Name           | contoso_contoso.timelinecontrol |
|                           |                                 | Managed by...  | -                               |
|                           |                                 | Owner          | -                               |
|                           |                                 | Status         | -                               |

[Load the next 100 items](#)

**Add**

**Cancel**

- Click **Publish All Customizations** and wait for the publishing to complete.

## 26 Exercise #3: Promote to production

**Objective:** In this exercise, you will export the Permit Management solution from your Dev environment and import it into your Production environment.

### 26.1 Task #1: Export Solution

1. Export Permit Management managed solution

- Log on to [Power Apps maker portal](#) and make sure you are in the **Dev** environment.
- Select **Solution**.
- Select the **Permit Management** solution and click **Export**.

The screenshot shows the 'Solutions' section of the Microsoft Power Apps portal. The 'Export' button in the top navigation bar is highlighted with a red arrow. The table below lists three solutions: 'PowerAppsTools\_contoso', 'Permit Management', and 'Common Data Services Default Solution'. The 'Permit Management' solution is selected.

| Display name                          | Created   | Version | Managed ex |
|---------------------------------------|-----------|---------|------------|
| PowerAppsTools_contoso                | 8/30/2019 | 1.0     | ⋮          |
| Permit Management                     | 8/22/2019 | 1.0.0.2 | ⋮          |
| Common Data Services Default Solution | 8/17/2019 | 1.0.0.0 | ⋮          |

- Click **Publish** and wait for the publishing to complete.

## Before you export X

### Publish all changes

If you made changes to this solution that you'd like to export, publish them now.

[Learn more](#)

**Publish**

### Check for issues

0 found on 08/26/2019

**Next**

**Cancel**

- Click **Next**.
- Select **Managed** and click **Export**.

## ← Export this solution ×

Version number\* ⓘ

Current version 1.0.0.2

1.0.0.3

Export as

Managed (recommended) ⓘ

The solution is moving to a test or production environment. [Learn more](#)

Unmanaged

The solution is moving to another development environment or source control. [Learn more](#)

**Export**

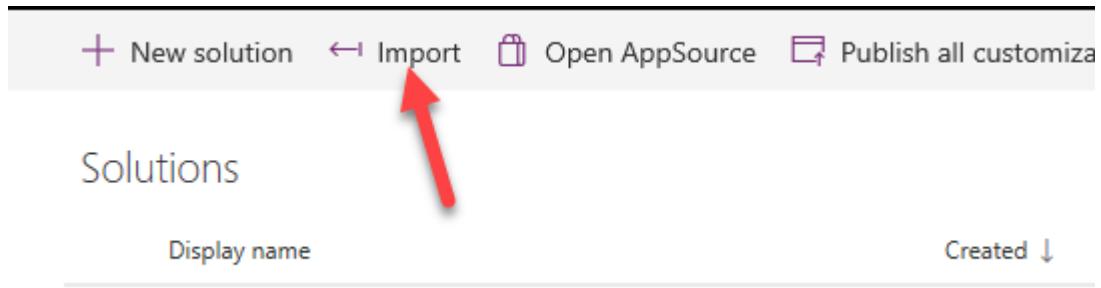
**Cancel**

- Save the **Exported** solution on your machine.
2. Export Permit Management unmanaged solution
- Select **Solution** again.
  - Select the **Permit Management** solution and click **Export**.
  - Click **Next**.
  - Select **Unmanaged**, edit the version number to match the Managed Solution you just exported and click **Export**.
  - Save the **Exported** solution on your machine.

## 26.2 Task #2: Import Solution

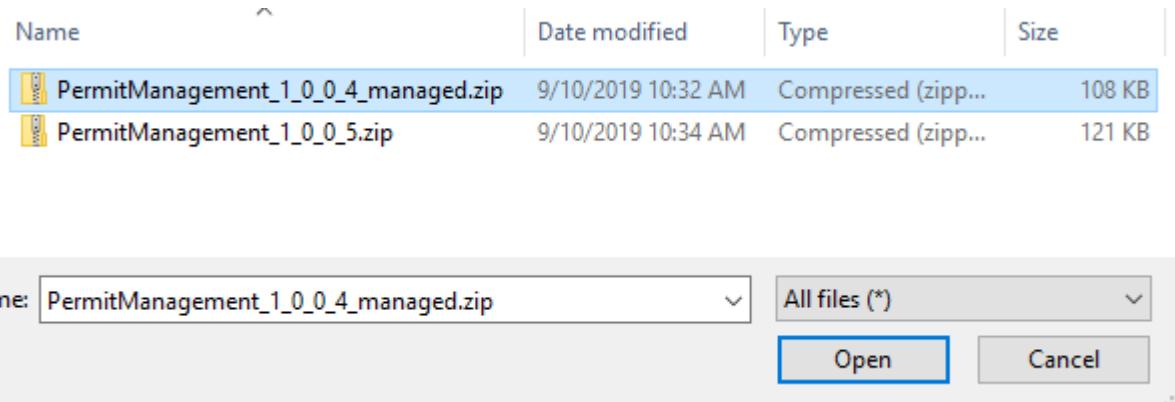
1. Import Permit Management managed solution

- Log on to [Power Apps maker portal](#) and make sure you are in the **Prod** environment.
- Select **Solution**.
- Click **Import**.



- Click **Choose File**.

- Select the **Managed** solution you exported and click **Open**.



- Click **Next**.
- Click **Next** again.
- Select the **Upgrade** option if you already have another version of the same solution.
- Click **Import**.

**Import Options** Help

**!** This solution package contains an update for a solution that is already installed.

**Solution Action** ([Learn more](#))

**Upgrade (recommended)**  
This option upgrades your solution to the latest version and rolls up all previous patches in one step. Any components associated to the previous solution version that are not in the newer solution version will be deleted.

**Stage for Upgrade**  
This option upgrades your solution to the higher version, but defers the deletion of the previous version and any related patches until you apply a solution upgrade later.

**Update (not recommended)**  
This option replaces your solution with this version. Components that are not in the newer solution will not be deleted and will remain in the system.

**Previous customizations on components included in this solution** ([Learn more](#))

[Back](#) [Import](#) [Cancel](#)

Wait for the import to complete and click **Close**

## 26.3 Review the production application by adding a few records and testing your progress.

### 26.4 lab: title: 'Lab 06: Plug-ins'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *Table* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

## 26.5 Lab 06 – Plug-ins

## 27 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will build two plugins. The first plugin will run when a new permit record is created, and it will check that there are no other permits that exists for the build sites that are “locked”. If a locked permit is found, the plugin will block the creation of this new permit. The second plugin will hook up and run when the Lock Permit custom action is invoked. In the prior module we defined the custom action, and now with the plugin step registered on execution of the custom action, it will now perform the lock permit business logic.

## 28 High-level lab steps

As part of building the plugins, you will complete the following activities.

- Create two plugins using a common provided base class
- Implement logic to work with a plugin registered on an Table event
- Implement logic to work with a plugin registered on a custom action event
- Deploy the plugins and associate them with your solution
- Use the plugin trace log to see traces from the plugin
- Debug the plugin on your local computer

### 28.1 Things to consider before you begin

- Do we know what events will trigger our plugins?
- Could what we are doing with the plugin, be done using Power Automate?
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production soon.

## 29 Exercise #1: Block New Permit Creation Plugin

**Objective:** In this exercise, you will create a plugin that will run on create permit. This plugin will check if there are any locked permits for the selected build site of the new permit and block the creation of the new permit.

### 29.1 Task #1: Create the plugin

1. Create a Visual Studio project
  - Start **Visual Studio**.
  - Click **File > New > Project**.
  - Select **Class Library (.NET Framework)** and click **Next**. Make sure you have selected the one with C#.

C# Windows Library UWP

Class Library (.NET Framework)  
A project for creating a C# class library (.dll)  
C# Windows Library

Class Library (.NET Core)  
A project for creating a class library that targets .NET Core.  
C# Windows Linux macOS Library

Class Library (.NET Standard)  
A project for creating a class library that targets .NET Standard.  
Visual Basic Android iOS Linux macOS Windows Library

Class Library (.NET Framework)  
A project for creating a VB class library (.dll)  
Visual Basic Windows Library

[Next](#)

- Enter **ContosoPackageProject** for **Project Name**, select a location to save the project, select **.NET Framework 4.6.2** for **Framework**, and then select **Create**.

## Configure your new project

Class Library (.NET Framework) C# Windows Library

Project name

Location  
[...](#)

Solution name [i](#)

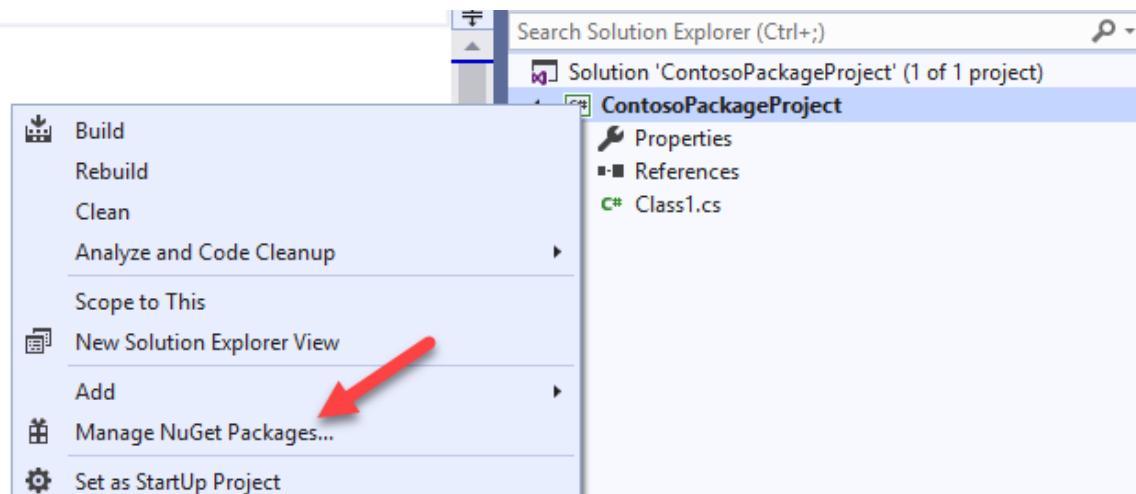
Place solution and project in the same directory

Framework

[Back](#) [Create](#)

## 2. Add NuGet packages

- Right click on the project name and select **Manage NuGet Packages**.



- Select the **Browse** tab.
- Search for **microsoft.CrmSdk.CoreAssemblies**.
- Select **Microsoft.crm sdk.core assemblies** (one authored by Microsoft) and click **Install**.

NuGet Package Manager: ContosoPackageProject

Package source: nuget.org

**Microsoft.CrmSdk.CoreAssemblies** by Microsoft 2.35M downloads  
Core assemblies required to develop managed code applications that can access the Microsoft Dynamics 365 web services.

**Microsoft.Extensions.Logging.Abstractions** by Microsoft, 140M downloads  
Logging abstractions for Microsoft.Extensions.Logging.

**Microsoft.Extensions.Configuration.Abstractions** by Microsoft, 137M downloads  
Abstractions of key-value pair based configuration.

**Microsoft.Extensions.Primitives** by Microsoft 141M downloads

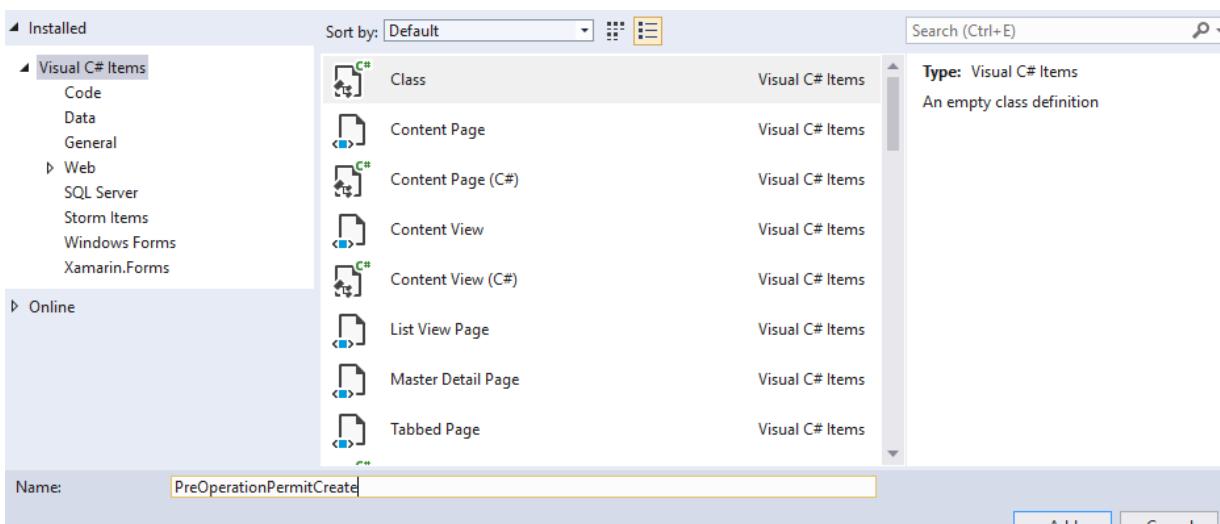
Version: Latest stable 9.0.2.17    Install

Description  
This package contains the official Microsoft.Xrm.Sdk.dll and Microsoft.Crm.Sdk.Proxy.dll assemblies plus tools and has been authored by the Microsoft Common Data Service SDK team.

- Read the license terms and then select **I agree** if you agree.

### 3. Delete Class1 and create new class file for the plugin

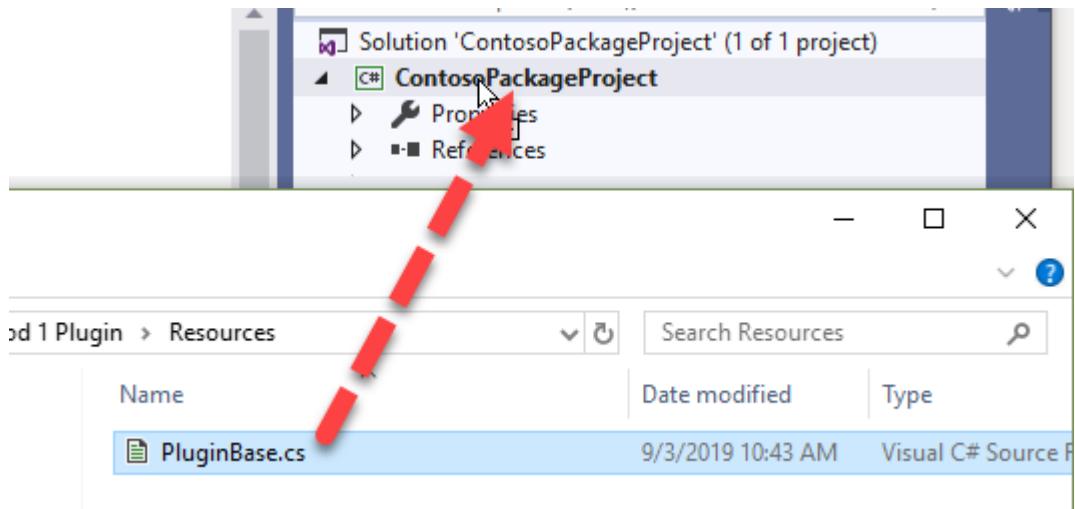
- Right click on **Class1.cs** and **Delete**. Click **OK** on the warning to delete this permanently.
- Right on the project and select **Add > Class**.
- Enter **PreOperationPermitCreate** for Name and click **Add**.



### 4. Add **PluginBase** file to the project.

- Locate the **PluginBase** file in the lab resources folder.

- Drag the **PluginBase** file to your Visual Studio project. Make sure that the file now exists in the folder structure.



- Review the **PluginBase** class. This class inherits from **IPlugin**.

```
using System.ServiceModel;
using Microsoft.Xrm.Sdk;

/// <summary>
/// Base class for all Plug-in classes.
/// </summary>
0 references
public abstract class PluginBase : IPlugin
{
 /// <summary>
 /// Plug-in Context object.
 /// </summary>
5 references
```

Close the **PluginBase** file.

- Add **using statements** to the **PreOperationPermitCreate** class, make the class **public**, and inherit from **PluginBase**

- Add the using statement below to the **PreOperationPermitCreate** class.

```
using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;
using ContosoPackageProject;
```

- Make the **PreOperationPermitCreate** public and inherit from **PluginBase**.

```
using System.Threading.Tasks;

using Microsoft.Xrm.Sdk;
using Microsoft.Xrm.Sdk.Query;

namespace ContosoPackageProject
{
 0 references
 public class PreOperationPermitCreate : PluginBase
 {
 }
}
```

- Override the **ExecuteCDSPlugin** method and get the Target Table and the Build Site Table reference.

- To override the **ExecuteCDSPlugin** method, add the code below to the **PreOperationPermitCreate** method.

```
protected override void ExecuteCDSPlugin(LocalPluginContext localcontext)
{
 base.ExecuteCDSPlugin(localcontext);
}
```

- To get the **Target** Table, add the code below inside the **ExecuteCDSPlugin** method.

```
var permitEntity = localcontext.PluginExecutionContext.InputParameters["Target"] as Entity;
```

- To get the Build Site Table reference, add the below code after **permitEntity** variable definition.

```
var buildSiteRef = permitEntity["contoso_buildsite"] as EntityReference;
```

- To add Trace Messages, add the below mentioned code after **buildSiteRef** variable definition.

```
localcontext.Trace("Primary Entity Id: " + permitEntity.Id);
```

```
localcontext.Trace("Build Site Entity Id: " + buildSiteRef.Id);
```

7. Create Fetch xml and that will get the count of locked permits matching the build site id and call retrieve multiple.

- Create the **FetchXML** string.

```
string fetchString = "<fetch output-format='xml-platform' distinct='false' version='1.0'"
```

- Call RetrieveMultiple and add Trace Message.

```
localcontext.Trace("Calling RetrieveMultiple for locked permits");
```

```
var response = localcontext.OrganizationService.RetrieveMultiple(new FetchExpression(fetchString));
```

8. Get the locked Permit Count and throw **InvalidPluginExecutionException** if the **Count** is more than 0

- Get the locker permits **Count**.

```
int lockedPermitCount = (int)((AliasedValue)response.Tables[0]["Count"]).Value;
```

- Add Trace Message, check if the **Count** is more than 0 and throw **InvalidPluginExecutionException** if it is more than 0.

```
localcontext.Trace("Locket Permit count : " + lockedPermitCount);
if (lockedPermitCount > 0)
{
 throw new InvalidPluginExecutionException("Too many locked permits for build site");
}
```

- The **ExecuteCDSPlugin** method should now look like the image below.

```

3 references
protected override void ExecuteCDSPlugin(LocalPluginContext localcontext)
{
 base.ExecuteCDSPlugin(localcontext);

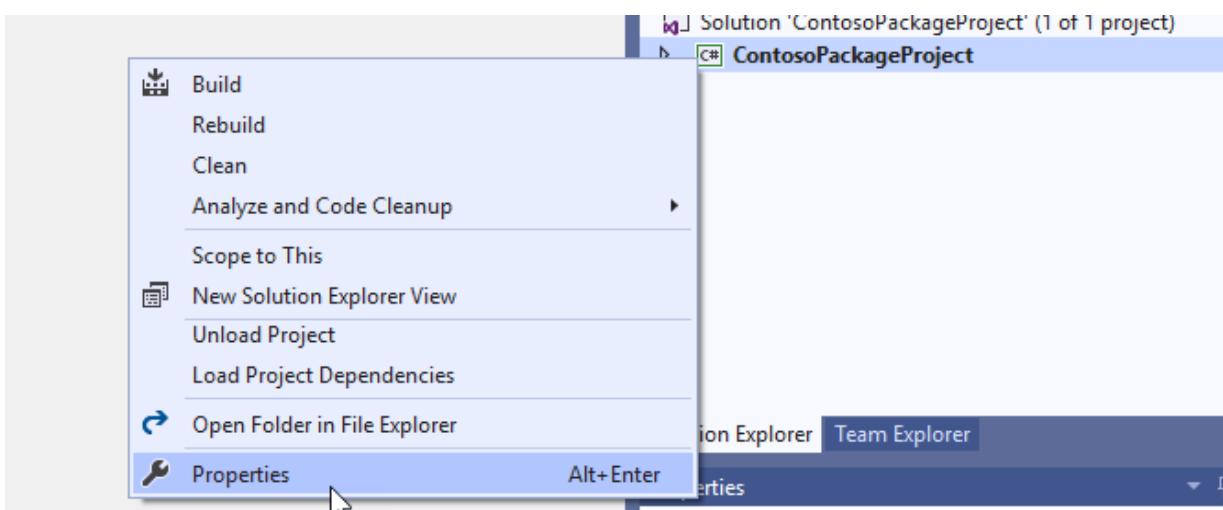
 var permitEntity = localcontext.PluginExecutionContext.InputParameters["Target"] as Entity;
 var buildSiteRef = permitEntity["contoso_buildsite"] as EntityReference;
 localcontext.Trace("Primary Entity Id: " + permitEntity.Id);
 localcontext.Trace("Build Site Entity Id: " + buildSiteRef.Id);
 string fetchString = "<fetch output-format='xml-platform' distinct='false' version='1.0' mapping='";
 localcontext.Trace("Calling RetrieveMultiple for locked permits");
 var response = localcontext.OrganizationService.RetrieveMultiple(new FetchExpression(fetchString));
 int lockedPermitCount = (int)((AliasedValue)response.Entities[0]["Count"]).Value;
 localcontext.Trace("Locket Permit count : " + lockedPermitCount);
 if (lockedPermitCount > 0)
 {
 throw new InvalidPluginExecutionException("Too many locked permits for build site");
 }
}

```

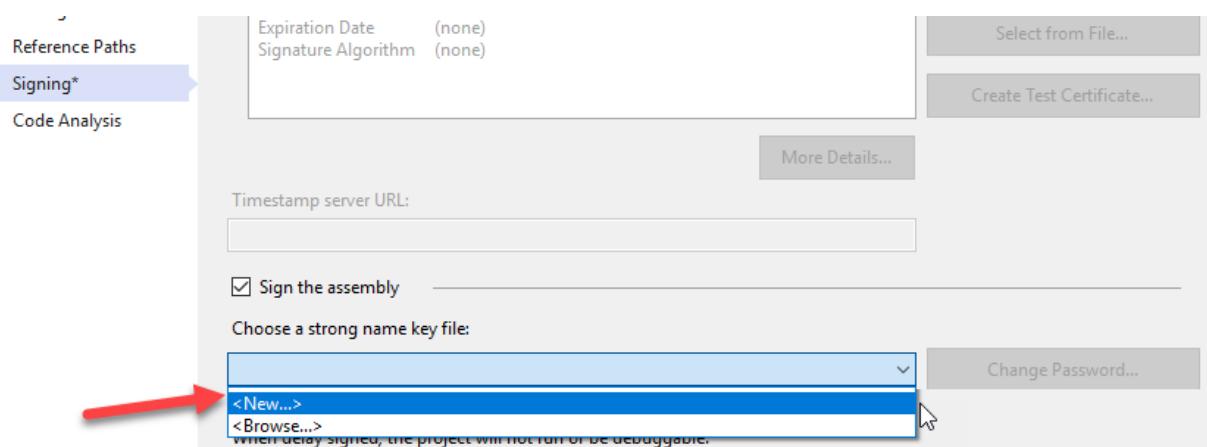
- Build the project and make sure it succeeds. To build the project, right click on the project and select **Build**. Check the output and make sure that the build is succeeded. If it does not, go back and review your work compared the steps documented here.

## 29.2 Task #2: Deploy the plugin

1. Create strong name key.
  - Right click on the **Project** and select **Properties**.

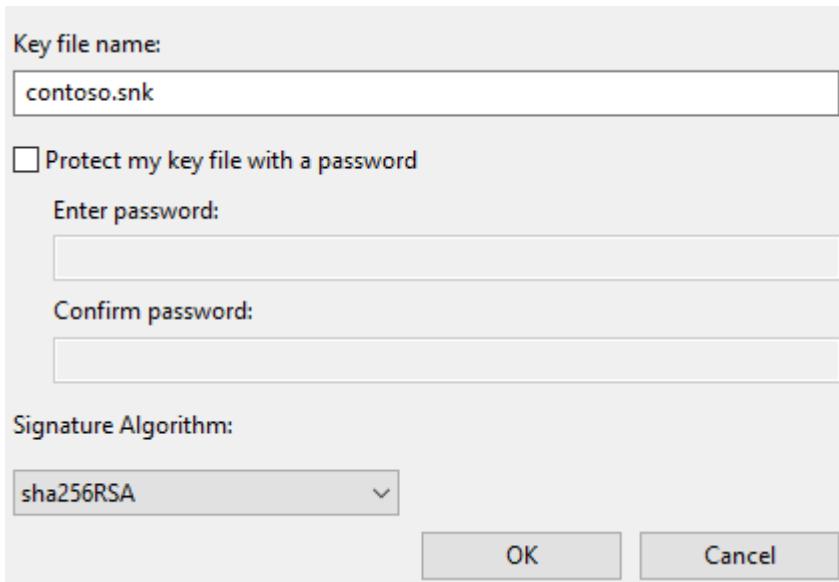


- Select the **Signing** tab, check the **Sign the assembly** checkbox and select <New...>.



- Enter **contoso.snk** for **Name**, uncheck the **Protect with a Password** checkbox, and click **OK**. Note: In case you get an access denied while creating the signature, close Visual Studio and run it

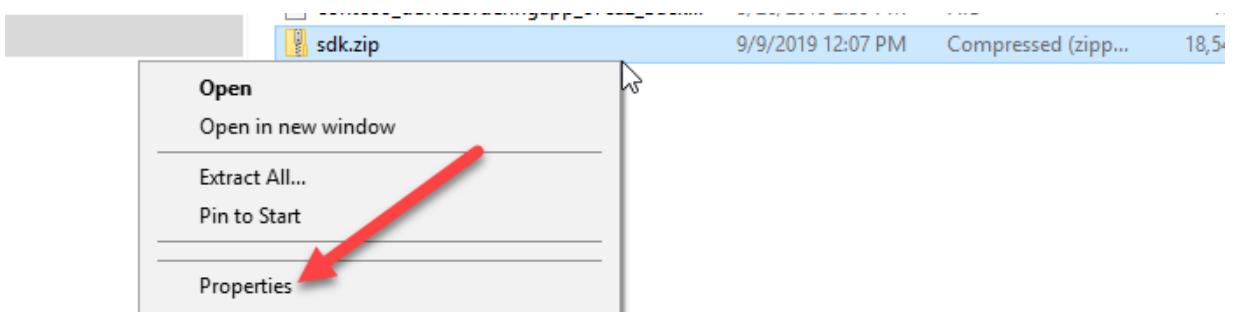
in administrator mode to successfully complete this step.



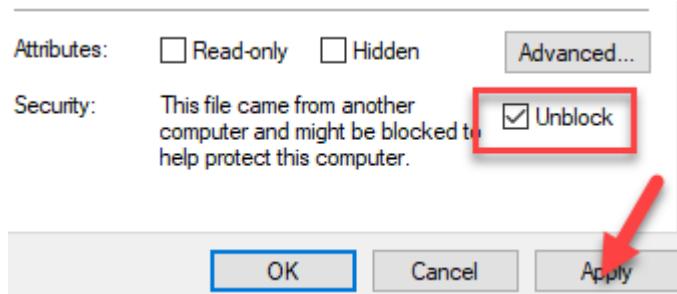
- Rebuild the project. Make sure the project is rebuilt successfully.
2. If you don't have CDS/Dynamics 365 SDK tools downloaded already, download them using the following method:
- Navigate to <https://xrm.tools/SDK>
  - Click **Download SDK Zip File**.

The page title is 'Download SDK Tools Zip'. Below it is a section titled 'Download a single Zip File' with the sub-instruction: 'This zip file contains Core Tools, Plugin Registration Tool, and Package Deployer runtime'. A large blue button labeled 'Download SDK Zip File' is centered. To the right, there is a note: 'After you download, you must right-click properties and unblock the file before you unzip it locally'.

- Save the zip file on your machine.
- Right click on the downloaded **sdk.zip** file and select **Properties**.



- Check the **Unblock** checkbox and click **Apply**.



- Click **OK**.
- Right click on the  **sdk.zip** file again and select **Extract All**.
- Complete extracting.

3. Start the plugin registration tool and sign in.

- Open the  **sdk** folder you extracted and click to open the **PluginRegistration** folder.
- Locate **PluginRegistration.exe** and double click to start. This will open a new window.

| Name                          | Date modified     | Type                  | Size   |
|-------------------------------|-------------------|-----------------------|--------|
| PluginProfiler.Plugins.dll    | 4/24/2018 6:30 PM | Application extens... | 78 KB  |
| PluginProfiler.Solution.zip   | 4/24/2018 6:30 PM | Compressed (zipp...   | 47 KB  |
| PluginRegistration.exe        | 4/24/2018 6:30 PM | Application           | 282 KB |
| PluginRegistration.exe.config | 4/24/2018 6:30 PM | XML Configuratio...   | 6 KB   |

4. Connect to your org.

- Click **Create New Connection**.



- Select **Office 365** and check the **Display List of available organization** and **Show Advanced** checkboxes. Select **Online Region** where your organization is located. If you are unsure what region to select, select **Don't Know**.
- Provide your **Microsoft Dataverse** credentials and click **Login**.

## Login

Deployment Type:  On-premises  Office 365

Sign in as current user  
 Display list of available organizations  
 Show Advanced

Advanced

|               |            |
|---------------|------------|
| Online Region | Don't Know |
| User Name     | [REDACTED] |
| Password      | *****      |

**Login** **Cancel**

- Select the **Dev** environment and click **Login**.

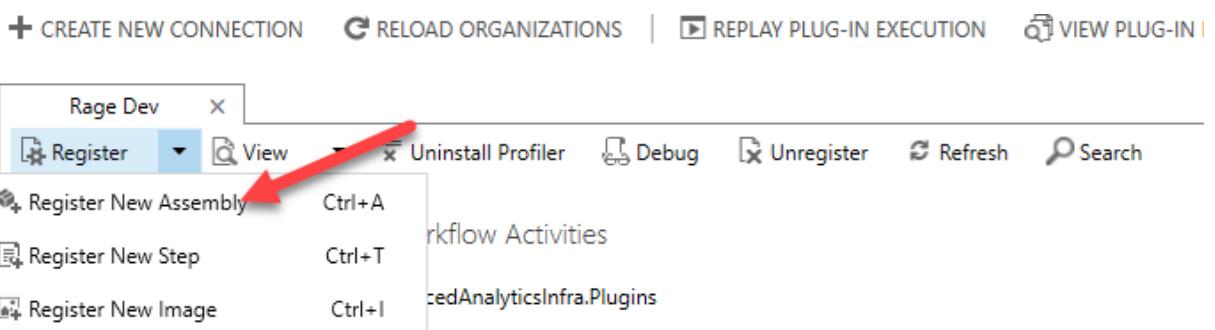
Multiple CRM Organizations are available to you. Please choose the CRM Organization you would like to connect to.

| CRM Region    | Organization Name       |
|---------------|-------------------------|
| North America | Rage Dev - org8a78478f  |
| North America | Rage Prod - org00da204e |

**Login** **Cancel**

#### 4. Register new assembly

- Click **Register** and select **Register New Assembly**.



- Click ... to browse.

# Register New Assembly

Step 1: Specify the location of the assembly to analyze

...  


Step 2: Select the plugin and workflow activities to register

Select All / Deselect All

- Browse to the bin/debug folder of your plugin project (**ContosoPackageProject**), select the **ContosoPackageProject.dll** file and click **Open**.

**Path:** PathToFolder/ContosoPackageProject/ContosoPackageProject/bin/Debug

| Name                        | Date modified      | Type                  | Size   |
|-----------------------------|--------------------|-----------------------|--------|
| ContosoPackageProject.dll   | 9/4/2019 11:41 AM  | Application extens... | 11 KB  |
| Microsoft.Crm.Sdk.Proxy.dll | 8/21/2019 12:47 AM | Application extens... | 288 KB |
| Microsoft.Xrm.Sdk.dll       | 8/21/2019 12:47 AM | Application extens... | 556 KB |



- Click **Register Selected Plugins**.

Step 2: Select the plugin and workflow activities to register

Select All / Deselect All

|                                     |                                                                                                                                                        |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> |  (Assembly) ContosoPackageProject                                     |
| <input checked="" type="checkbox"/> |  (Plugin) ContosoPackageProject.PreOperationPermitCreate - Isolatable |

Step 3: Specify the isolation mode

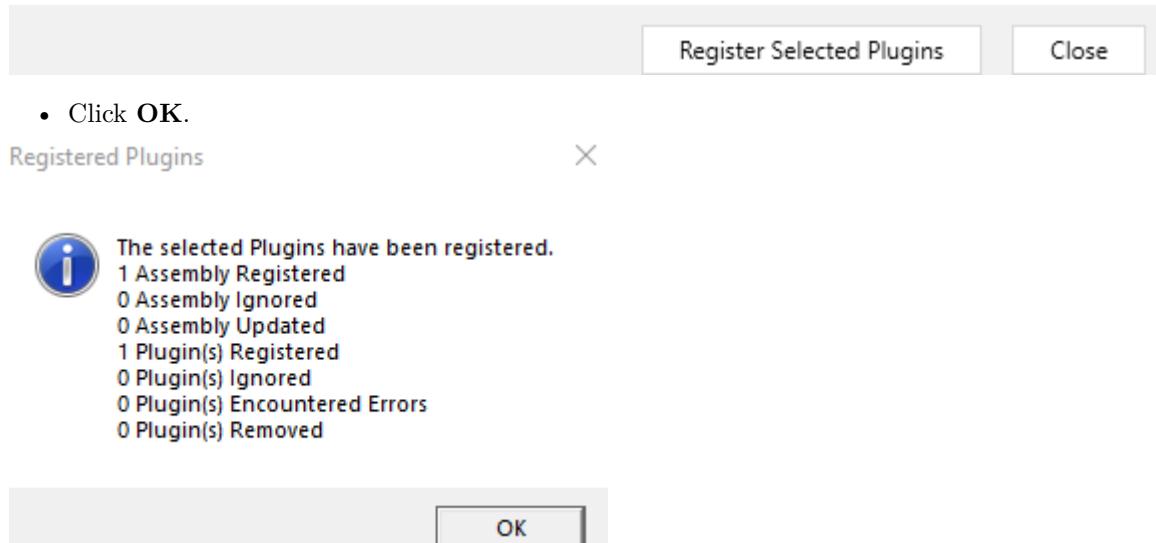
- Sandbox [?](#)  
 None

Step 4: Specify the location where the assembly should be stored

- Database [?](#)  
 Disk [?](#)  
 GAC [?](#)

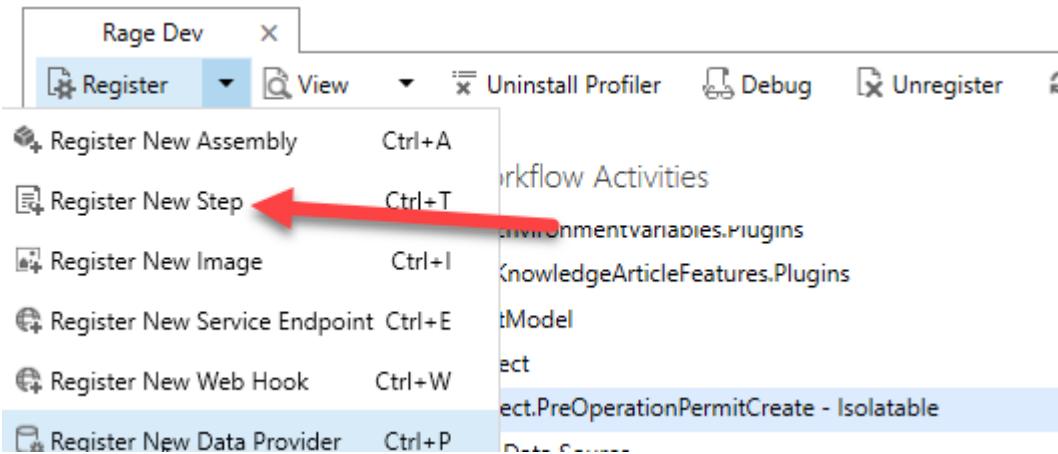
Step 5: Log

|  |
|--|
|  |
|--|



5. Register new step

- Select the assembly you just registered.
- Click **Register** and select **Register New Step**.



- Enter **Create** for Message.
- Enter **contoso\_permit** for Primary Table.
- Select **PreOperation** from dropdown for Event Pipeline Stage of Execution and click Register New Step.

### Register New Step

**General Configuration Information**

|                       |                                                                                              |
|-----------------------|----------------------------------------------------------------------------------------------|
| Message               | <input type="text" value="Create"/>                                                          |
| Primary Entity        | <input type="text" value="contoso_permit"/>                                                  |
| Secondary Entity      | <input type="text"/>                                                                         |
| Filtering Attributes  | <input type="text" value="Message does not support Filtered Attributes"/>                    |
| Event Handler         | <input type="text" value="(Plugin) ContosoPackageProject.LockPermitCancelInspections"/>      |
| Step Name             | <input type="text" value="ContosoPackageProject.LockPermitCancelInspections: Create of cc"/> |
| Run in User's Context | <input type="text" value="Calling User"/>                                                    |
| Execution Order       | <input type="text" value="1"/>                                                               |
| Description           | <input type="text" value="ContosoPackageProject.LockPermitCancelInspections: Create of cc"/> |

**Unsecure Configuration**

**Secure Configuration**

Delete AsyncOperation if StatusCode = Successful

- Step should now be registered in the assembly plugin.
- ▶ **(Assembly) Microsoft.Crm.ObjectModel**
  - ◀ **(Assembly) ContosoPackageProject**
  - └ **(Plugin) ContosoPackageProject.PreOperationPermitCreate - Isolatable**
  - └ **(Step) ContosoPackageProject.PreOperationPermitCreate: Create of contoso\_permit**
  - ▶ **(Data Source) Component Layer Data Source**

## 30 Exercise #2: Create Custom Action Plugin

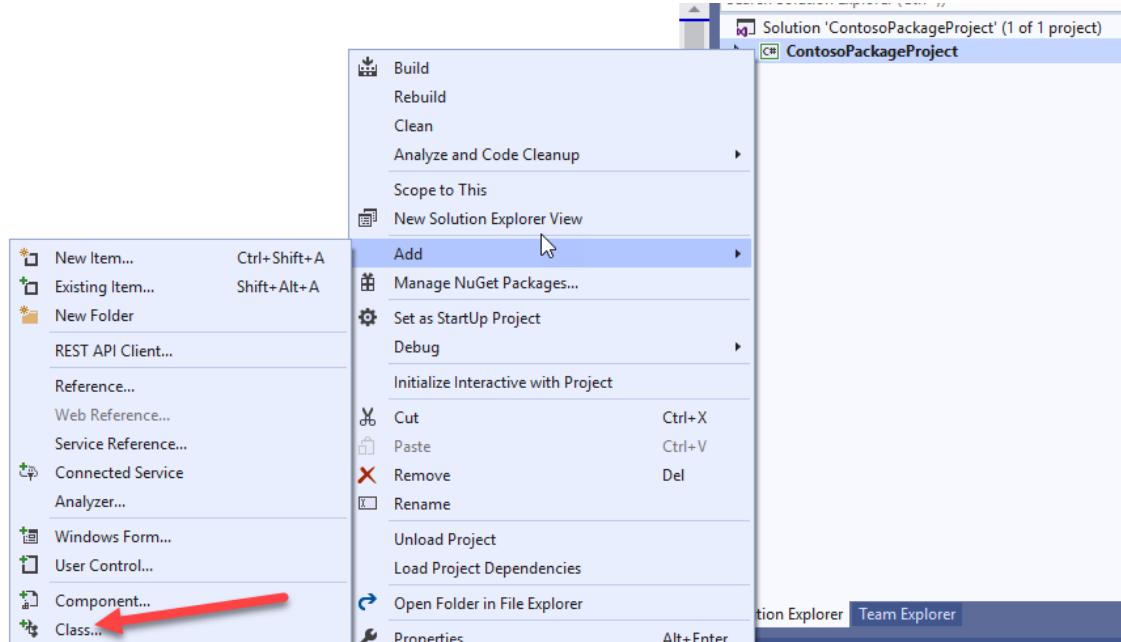
**Objective:** In this exercise, you will create and register a plugin that will be invoked when the lock permit custom action is used. This plugin will be used to implement the business logic of locking the permit. Specifically, it will update the permit to indicate it is locked and then cancel any pending inspections.

**Note:** If you did not create the custom action in a prior lab, look in your resources folder for how to add it here before you proceed.

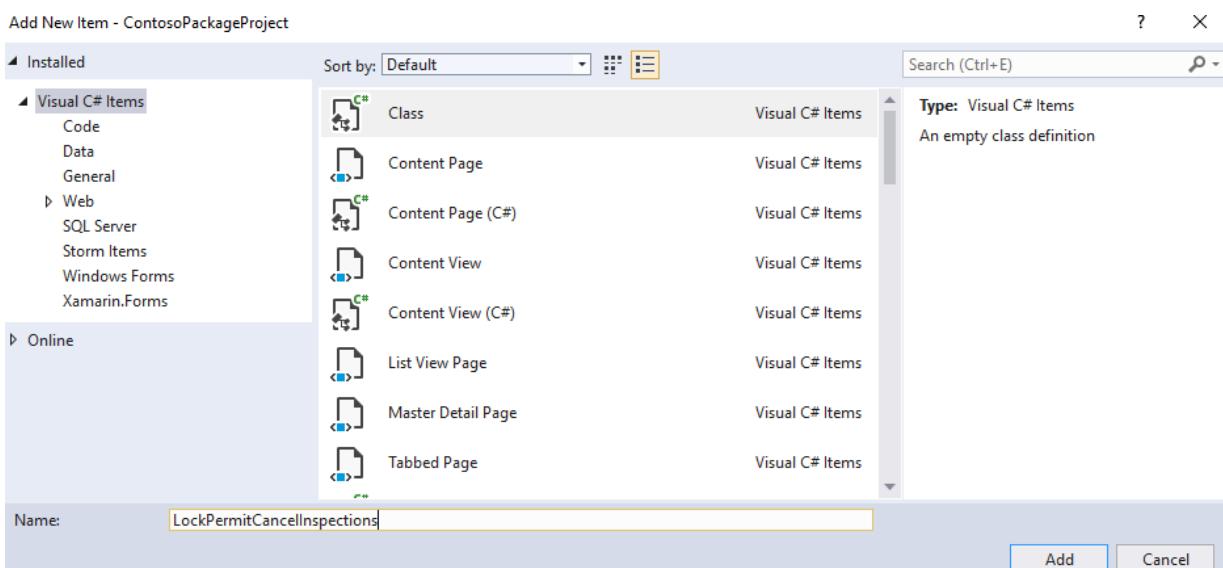
### 30.1 Task #1: Add a new plugin to the project

1. Add new class to the project and name it **LockPermitCancelInspections**

- Right on the project and select **Add > Class**.



- Enter **LockPermitCancelInspections** for Name and click **Add**.



2. Add **using** statements to the **LockPermitCancelInspections** class, make the class **public**, and inherit from **PluginBase**

- Add the using statement below to the **LockPermitCancelInspections** class.

```
using Microsoft.Xrm.Sdk;
using System.Text.RegularExpressions;
using ContosoPackageProject;
using Microsoft.Xrm.Sdk.Query;
```

- Make the **LockPermitCancelInspections** public and inherit from **PluginBase**.

```

4 using System.Text;
5 using System.Threading.Tasks;
6
7 using Microsoft.Xrm.Sdk;
8 using System.Text.RegularExpressions;
9
10
11 namespace ContosoPackageProject
12 {
13 public class LockPermitCancelInspections : PluginBase
14 {
15 }
16 }
17
18

```

3. To override the ExecuteCDSPlugin method and get the reason value from the input parameter.

- Override the **ExecuteCDSPlugin** method. Add the code below inside the **LockPermitCancelInspections** method.

```

protected override void ExecuteCDSPlugin(LocalPluginContext localcontext)
{
 base.ExecuteCDSPlugin(localcontext);
}

```

4. Get the target Table reference, Table, set status reason to lock, and update the permit record.

- Get the target **Table Reference** and **Table**.

```

var permitEntityRef = localcontext.PluginExecutionContext.InputParameters["Target"] as EntityReference;
Entity permitEntity = new Entity(permitEntityRef.LogicalName, permitEntityRef.Id);

```

- Add **Trace** message and Set the **Status Reason** to **Lock**. 463270000 is the lock value of the Status Reason option-set and statuscode is the name of the status reason Column.

```

localcontext.Trace("Updating Permit Id : " + permitEntityRef.Id);
permitEntity["statuscode"] = new OptionSetValue(463270000);

```

- Update the **Permit** record and add **Trace** message.

```

localcontext.OrganizationService.Update(permitEntity);
localcontext.Trace("Updated Permit Id " + permitEntityRef.Id);

```

```

5 references
protected override void ExecuteCDSPlugin(LocalPluginContext localcontext)
{
 base.ExecuteCDSPlugin(localcontext);
 var permitEntityRef = localcontext.PluginExecutionContext.InputParameters["Target"] as EntityReference;
 Entity permitEntity = new Entity(permitEntityRef.LogicalName, permitEntityRef.Id);
 localcontext.Trace("Updating Permit Id : " + permitEntityRef.Id);
 permitEntity["statuscode"] = new OptionSetValue(463270000);
 localcontext.OrganizationService.Update(permitEntity);
 localcontext.Trace("Updated Permit Id " + permitEntityRef.Id);
}

```

## 30.2 Task #2: Get Related Inspections and Cancel

1. Create query and condition expressions.

- Create the **QueryExpression**. Add the code below to the **ExecuteCDSPlugin** method.

```

QueryExpressionqe = new QueryExpression();
qe.EntityName = "contoso_inspection";
qe.ColumnSet = new ColumnSet("statuscode");

```

- Create the **ConditionExpression**.

```

ConditionExpression condition = new ConditionExpression();
condition.Operator = ConditionOperator.Equal;
condition.AttributeName = "contoso_permit";
condition.Values.Add(permitEntityRef.Id);

• Set the Criteria of the query.

qe.Criteria = new FilterExpression(LogicalOperator.And);

• Add the ConditionExpression to the Criteria of the QueryExpression.

qe.Criteria.Conditions.Add(condition);

 localcontext.Trace("Updated Permit Id " + permitEntityRef.Id);

 QueryExpression qe = new QueryExpression();
 qe.EntityName = "contoso_inspection";
 qe.ColumnSet = new ColumnSet("statuscode");

 ConditionExpression condition = new ConditionExpression();
 condition.Operator = ConditionOperator.Equal;
 condition.AttributeName = "contoso_permit";
 condition.Values.Add(permitEntityRef.Id);

 qe.Criteria = new FilterExpression(LogicalOperator.And);
 qe.Criteria.Conditions.Add(condition);

}

}

```

2. Retrieve the inspections and iterate through the returned Tables.

- Retrieve the **Inspections** and add **Trace** messages.

```

localcontext.Trace("Retrieving inspections for Permit Id " + permitEntityRef.Id);
var inspectionsResult = localcontext.OrganizationService.RetrieveMultiple(qe);
localcontext.Trace("Retrieved " + inspectionsResult.TotalRecordCount + " inspection records");

```

- Create a **variable** that will keep track of the canceled **Inspections** count and Iterate through the returned Tables.

```

int canceledInspectionsCount = 0;
foreach (var inspection in inspectionsResult.Entities)
{
}

```

3. Retrieve the selected status reason option and check if it is set to new request or pending.

- Get the currently selected value of the **Status Reason** option-set. Add the code below inside the **foreach** loop.

```
var currentValue = inspection.GetAttributeValue<OptionSetValue>("statuscode");
```

- Check if the selected option is **New Request** or **Pending** and increment the count. 1 is the value of the New Request option and 463270000 id the value of the Pending option. This should be placed inside the foreach loop.

```

if (currentValue.Value == 1 || currentValue.Value == 463270000)
{
 canceledInspectionsCount++;
}

```

4. Cancel the inspections that are pending or new request

- Set the **Status Reason** selected value to **Canceled**. Add the code below inside the if statement inside the foreach loop. Make sure that 463270003 is the value for **Canceled Status Reason** in the

**Inspections** Table. If this differs, please update the value with actual value for **Canceled Status Reason**.

```

 inspection["statuscode"] = new OptionSetValue(463270003);

• Update the Inspection and add Trace messages.

 localcontext.Trace("Canceling inspection Id : " + inspection.Id);
 localcontext.OrganizationService.Update(inspection);
 localcontext.Trace("Canceled inspection Id : " + inspection.Id);

 qe.Criteria.Conditions.Add(condition);

 localcontext.Trace("Retrieving inspections for Permit Id " + permitEntityRef.Id);
 var inspectionsResult = localcontext.OrganizationService.RetrieveMultiple(qe);
 localcontext.Trace("Retrieved " + inspectionsResult.TotalRecordCount + " inspection records");

 int canceledInspectionsCount = 0;
 foreach (var inspection in inspectionsResult.Entities)
 {
 var currentValue = inspection.GetAttributeValue<OptionSetValue>("statuscode");

 if (currentValue.Value == 1 || currentValue.Value == 463270000)
 {
 canceledInspectionsCount++;

 inspection["statuscode"] = new OptionSetValue(463270003);
 localcontext.Trace("Canceling inspection Id : " + inspection.Id);
 localcontext.OrganizationService.Update(inspection);
 localcontext.Trace("Canceled inspection Id : " + inspection.Id);
 }
 }
}

```

### 30.3 Task #3: Set Output Parameter and Create Note Record

1. Check if at least one Inspection was canceled and CanceledInspectionsCount output Parameter.

- Check if at least one **Inspection** was canceled. Add the code below after the **foreach** loop.

```

 if (canceledInspectionsCount > 0)
 {

 }

```

- Set the **CanceledInspectionsCount** output parameter. Add the code below inside the if statement outside the foreach loop.

```

 localcontext.PluginExecutionContext.OutputParameters["CanceledInspectionsCount"] = canceledI
 }

 if (canceledInspectionsCount > 0)
 {
 localcontext.PluginExecutionContext.OutputParameters["CanceledInspectionsCount"] = canceledInspectionsCount + " Inspections were canceled";
 }
}

```

2. Check if the Input Parameters contain Reason and Create the Note record.

- Check if **Reason** contains in the **InputParameters**. Add the code below after the last if statement.

```

 if (localcontext.PluginExecutionContext.InputParameters.ContainsKey("Reason"))
 {

 }

```

- Build the **Note** record and add **Trace Message**. Add the code below inside the if statement.

```

 localcontext.Trace("building a note record");
 Entity note = new Entity("annotation");
 note["subject"] = "Permit Locked";
 note["notetext"] = "Reason for locking this permit: " + localcontext.PluginExecutionContext.

```

```

 note["objectid"] = permitEntityRef;
 note["objecttypecode"] = permitEntityRef.LogicalName;

 • Add Trace Message and create the Note record.

 localcontext.Trace("Creating a note reocord");
 var createdNoteId = localcontext.OrganizationService.Create(note);

 • Check if the Note record was created and add Trace Message.

 if (createdNoteId != Guid.Empty)
 localcontext.Trace("Note record was created");

 if (canceledInspectionsCount > 0)
 {
 localcontext.PluginExecutionContext.OutputParameters["CanceledInspectionsCount"] = canceledInspectionsCount + " Inspections were canceled";
 }

 if (localcontext.PluginExecutionContext.InputParameters.ContainsKey("Reason"))
 {
 localcontext.Trace("building a note reocord");
 Entity note = new Entity("annotation");
 note["subject"] = "Permit Locked";
 note["notetext"] = "Reason for locking this permit: " + localcontext.PluginExecutionContext.InputParameters["Reason"];
 note["objectid"] = permitEntityRef;
 note["objecttypecode"] = permitEntityRef.LogicalName;

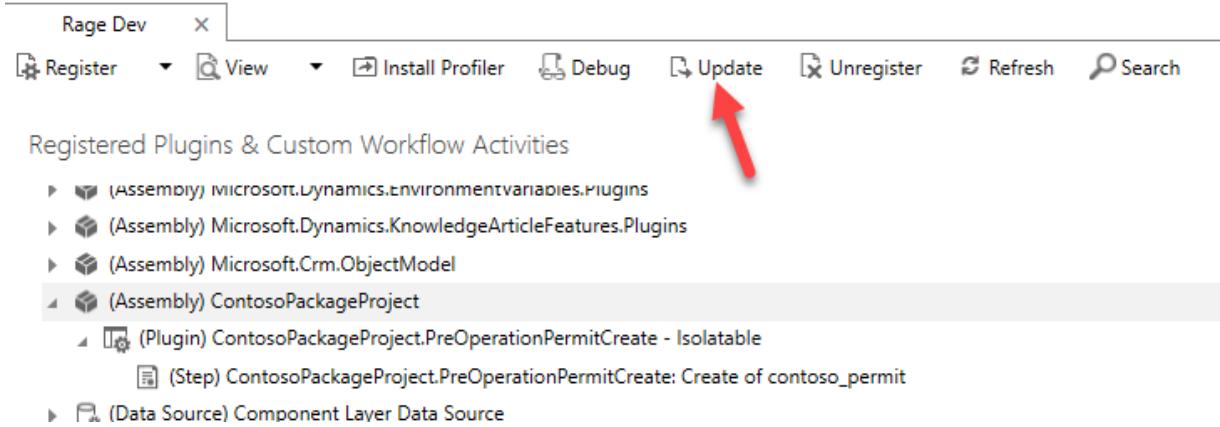
 localcontext.Trace("Creating a note reocord");
 var createdNoteId = localcontext.OrganizationService.Create(note);
 if (createdNoteId != Guid.Empty)
 localcontext.Trace("Note record was created");
 }
 }
}

```

3. Build plugin by right clicking on the project and select **Build** and make sure the build succeeds.

### 30.4 Task #4: Deploy Plugin

1. If you do not have the plugin registration tool running already, follow instructions in Exercise #1, Task #2 to run the tool and connect to the organization.
2. Update the assembly
  - Select **ContosoPackageProject** and click **Update**.



- Click ... to **Browse**.

Step 1: Specify the location of the assembly to analyze

 ... Load Assembly

Step 2: Select the plugin and workflow activities to register

Select All / Deselect All

-  (Assembly) ContosoPackageProject
-  (Plugin) ContosoPackageProject.PreOperationPermitCreate - Isolatable
-  (Step) ContosoPackageProject.PreOperationPermitCreate: Create of contoso\_permit

Step 3: Specify the isolation mode

- Browse to the **debug** folder of your plugin project, select the **ContosoPackageProject.dll** file and click **Open**.
- Check **Select All** checkbox and click **Update Selected Plugins**.

Update Assembly Properties and Content

Load Assembly

Step 2: Select the plugin and workflow activities to register

Select All / Deselect All

A screenshot of a software interface showing a list of registered assembly components. At the top left is a checkbox labeled '(Assembly) ContosoPackageProject' which is checked. Below it are two more items, also checked: '(Plugin) ContosoPackageProject.PreOperationPermitCreate - Isolatable' and '(Plugin) ContosoPackageProject.LockPermitCancelInspections - Isolatable'. There are navigation arrows at the bottom of the list.

Step 3: Specify the isolation mode

Sandbox [?](#)

None

Step 4: Specify the location where the assembly should be stored

Database [?](#)

Disk [?](#)

GAC [?](#)

Step 5: Log

A large, empty rectangular text area with a thin gray border, intended for logging information.

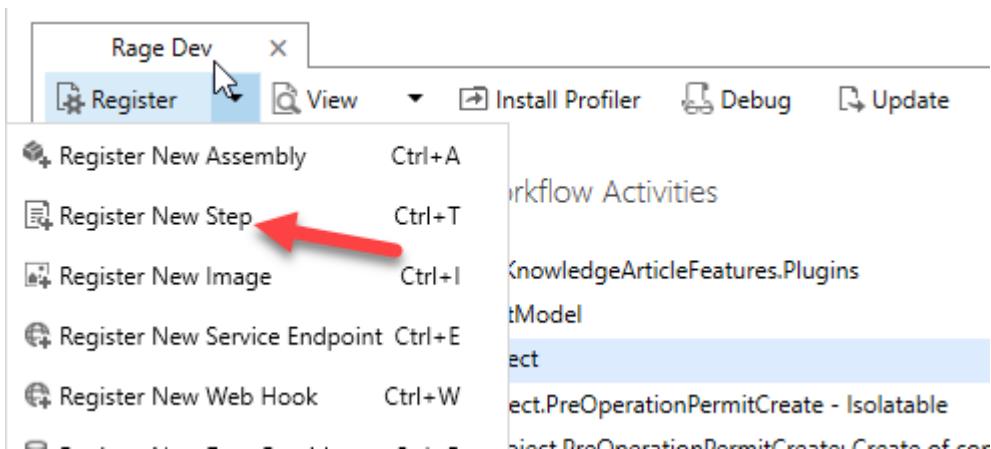
Update Selected Plugins

Close

- Click **OK**.

3. Register new step

- Select the assembly (ContosoPackageProject.LockPermitCancelInspections).
- Click **Register** and select **Register New Step**.



- Enter **contoso** in the **Message** textbox and select **contoso\_LockPermit**.

## Register New Step

### General Configuration Information

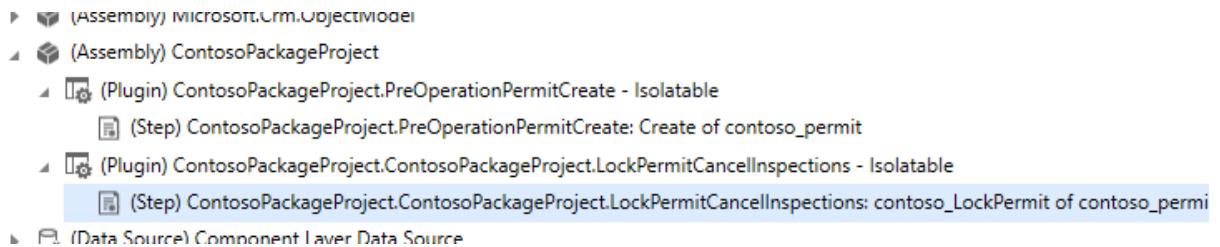
|                |                                                 |
|----------------|-------------------------------------------------|
| Message        | <input type="text" value="contoso_LockPermit"/> |
| Primary Entity | <input type="text" value="contoso_LockPermit"/> |

- Enter **contoso\_permit** for **Primary Table**.
- Make sure that Event Handler is selected for **LockPermitCancelInspections** plugin.
- Select **PreOperation** from dropdown for **Event Pipeline Stage of Execution** and click **Register New Step**.

## Register New Step

| General Configuration Information                                         |                                                                                           | Unsecure Configuration                                                                |
|---------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Message                                                                   | <input type="text" value="contoso_LockPermit"/>                                           |                                                                                       |
| Primary Entity                                                            | <input type="text" value="contoso_permit"/>                                               |                                                                                       |
| Secondary Entity                                                          | <input type="text"/>                                                                      |                                                                                       |
| Filtering Attributes                                                      | <input type="text" value="Message does not support Filtered Attributes"/>                 |                                                                                       |
| Event Handler                                                             | (Plugin) ContosoPackageProject.ContosoPackageProject.Lockf                                |                                                                                       |
| Step Name                                                                 | <input type="text" value="ContosoPackageProject.ContosoPackageProject.LockPermitCancel"/> |                                                                                       |
| Run in User's Context                                                     | <input type="text" value="Calling User"/>                                                 |                                                                                       |
| Execution Order                                                           | <input type="text" value="1"/>                                                            |                                                                                       |
| Description                                                               | <input type="text" value="ContosoPackageProject.ContosoPackageProject.LockPermitCancel"/> |                                                                                       |
| Event Pipeline Stage of Execution                                         |                                                                                           | Execution Mode                                                                        |
| PreOperation                                                              | <input type="radio"/>                                                                     | <input checked="" type="radio"/> Asynchronous                                         |
|                                                                           | <input type="radio"/>                                                                     | <input checked="" type="radio"/> Server                                               |
|                                                                           | <input type="radio"/>                                                                     | <input type="radio"/> Synchronous                                                     |
|                                                                           | <input type="radio"/>                                                                     | <input type="radio"/> Offline                                                         |
| <input type="checkbox"/> Delete AsyncOperation if StatusCode = Successful |                                                                                           |                                                                                       |
|                                                                           |                                                                                           | <input type="button" value="Register New Step"/> <input type="button" value="Close"/> |

- Step should now be registered in the assembly.



## 31 Exercise #3: Test Plugins

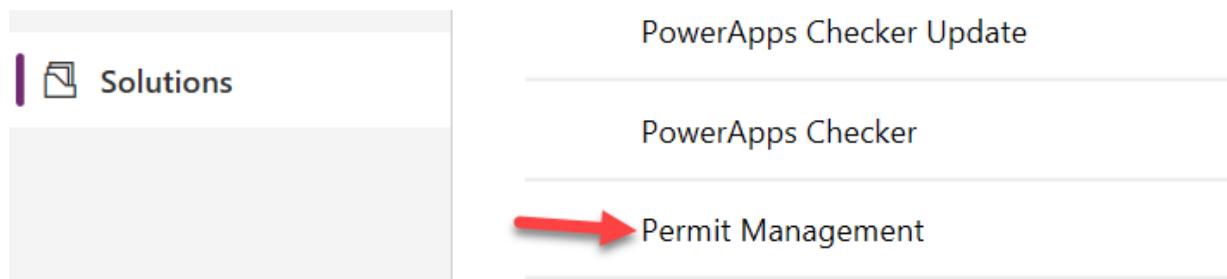
**Objective:** In this exercise, you will test the plugins you created.

### 31.1 Task #1: Test Lock Plugin

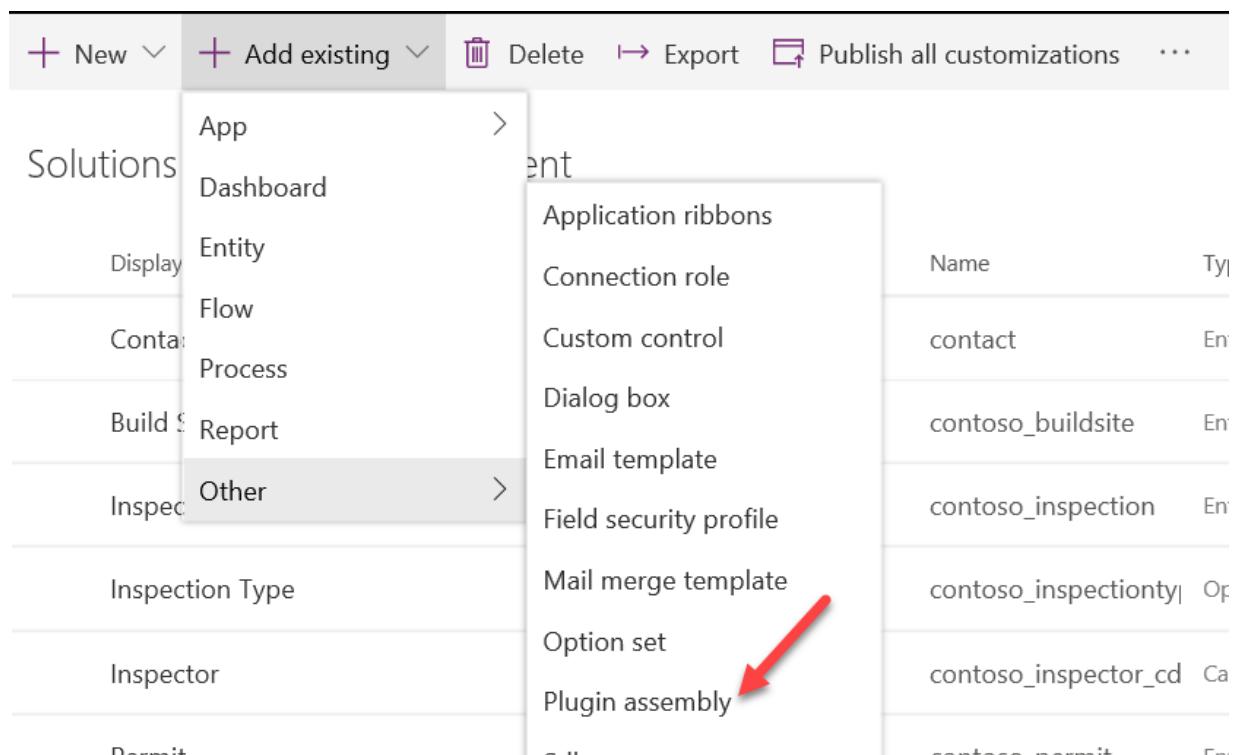
1. Add Plugin Assembly and Plugin - step to the Permit Management solution

**Note:** This is only required to be done after deploying it first time to ensure that when the project solution is moved to the production environments, the plugins and the registered steps are included.

- Sign in to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select **Solution** and click to open the **Permit Management** solution.



- Click **Add Existing | Other | Plugin Assembly**.



- Select **ContosoPackageProject** and click **Add**.

## 1 plugin assembly selected

| Name                                        | Version |
|---------------------------------------------|---------|
| ContosoPackageProject                       | 1.0.0.0 |
| Microsoft.Dynamics.ComponentHistory.Plugins | 9.0.0.0 |
| RWB2016.Plugins                             | 1.0.0.0 |

Add
Cancel

- Click **Add Existing | Other | Plug-in step.**

The screenshot shows the 'Permit Management' ribbon tab selected. A context menu is open, listing various options like Dialog, Entity, Flow, etc. A red arrow points to the 'Plug-in step' option in the list.

| Name                                                        | SDK message         | Event handler                   |
|-------------------------------------------------------------|---------------------|---------------------------------|
| ContosoPackageProject.ContosoPackageProject.LockPermitCa    | contoso_LockPermit  | ContosoPackageProject.Contoso   |
| ContosoPackageProject.PreOperationPermitCreate: Create of c | Create              | ContosoPackageProject.PreOper   |
| Microsoft.Xrm.DataProvider.Odata.V4.Plugins.ODataEntityData | Create              | Microsoft.Xrm.DataProvider.Odat |
| Microsoft.Xrm.DataProvider.Odata.V4.Plugins.ODataEntityData | Update              | Microsoft.Xrm.DataProvider.Odat |
| Ribbon Workbench Action                                     | rwb_CustomiseRibbon | RWB2016.Plugins.RibbonCustomi   |

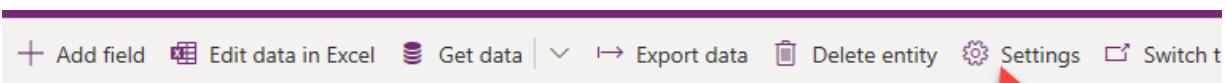
- Select both SDK Messages you created and click **Add**.

## 2 sdk messages selected

| Name                                                        | SDK message         | Event handler                   |
|-------------------------------------------------------------|---------------------|---------------------------------|
| ContosoPackageProject.ContosoPackageProject.LockPermitCa    | contoso_LockPermit  | ContosoPackageProject.Contoso   |
| ContosoPackageProject.PreOperationPermitCreate: Create of c | Create              | ContosoPackageProject.PreOper   |
| Microsoft.Xrm.DataProvider.Odata.V4.Plugins.ODataEntityData | Create              | Microsoft.Xrm.DataProvider.Odat |
| Microsoft.Xrm.DataProvider.Odata.V4.Plugins.ODataEntityData | Update              | Microsoft.Xrm.DataProvider.Odat |
| Ribbon Workbench Action                                     | rwb_CustomiseRibbon | RWB2016.Plugins.RibbonCustomi   |

Add
Cancel

- Now, open the **Permit** Table and click **Settings**.



Solutions > Permit Management > **Permit**

Fields Relationships Business rules Views Forms Dashboards Charts Keys Data

Display name ↑ ↴

Name ↴

- Check **Enable Attachments and Notes**, and then click **Done**.

## Edit entity

X

Display name \*

Permit

Plural display name \*

Permits

Name \* ⓘ

contoso\_ Permit

Enable attachments (including notes and file)

More settings ↴

Done

Cancel

- Click **Okay**.
- Finally, select **Save Table**.
- Select **Solutions** and click **Publish All Customizations**.

2. Start the Permit Management application and enable Plugin Tracing
  - Select **Apps**.
  - Click to open the **Permit Management** application.

The screenshot shows the Dynamics 365 navigation bar on the left with options like Home, Learn, Apps, Create, Data, Flows, and Chatbots. The Apps section is selected. To the right, the 'Apps' page is displayed with a table. The table has a header row with columns for Name, Inspector, and three dots. Below is a list of apps: Inspector, Permit Management (with a red arrow pointing to it), and Solution Health Hub.

| Name                | Inspector |     |
|---------------------|-----------|-----|
| Inspector           |           | ... |
| Permit Management   |           | ... |
| Solution Health Hub |           | ... |

- Click **Settings** and select **Advanced Settings**.

The screenshot shows the Dynamics 365 ribbon at the top with icons for search, refresh, new, filter, settings, and help. Below the ribbon, the 'Run Report' button is visible. On the left, there's a sidebar with 'Personalization Settings' and 'Advanced Settings' (which has a red arrow pointing to it). The main content area shows 'Toast Notification Display T...'.

- Click **Settings** and select **Administration**.

The screenshot shows the Dynamics 365 ribbon at the top with 'Dynamics 365', 'Settings', and 'Business Management'. Below the ribbon, the 'Settings' icon is highlighted. The main content area is divided into three tabs: Business, Customization, and System. The System tab is selected and has a red arrow pointing to the 'Administration' option under its sub-options (Business Management, Templates, Customizations, Solutions, Administration, Security, Email Configuration).

| Business               | Customization  | System         |
|------------------------|----------------|----------------|
| Business Management... | Customizations | Administration |
| Templates              | Solutions      | Security       |

- Click **System Settings**.
- Select **Customization** tab.
- Set Enable Plugin Logging to Plugin Trace Log to **All** and click **OK**.

**Application mode**

Set whether Microsoft Dynamics 365 can be opened in a browser window without menu, navigation, and command bars.

Open Microsoft Dynamics 365 in Application mode

**Plug-in and custom workflow activity tracing**

Enable logging to plug-in trace log

All

**Enable Microsoft Flow**

Show Microsoft Flow on forms and in the site map

Yes  No

3. Create test record

- Go back to the **Permit Management** application.
- Select **Inspections**.
- You should have two inspections one **Failed** and one **Passed**. If not, open them and update the records.
- Click **New**.

| Name                 | Status | Type            | Scheduled Date |
|----------------------|--------|-----------------|----------------|
| Electric Inspections | Failed | Initial Insp... | 9/19/2019      |
| Framing Inspection   | Passed | Initial Insp... | 8/30/2019      |

- Enter **Plumbing Inspection** for **Name**, select **Initial Inspection** for **Type**, select a permit, provide **Schedule Data**, select **Pending** for **Status Reason**, and click **Save**.



General Related

|                 |                                                     |
|-----------------|-----------------------------------------------------|
| Name            | * <b>Plumbing Inspections</b>                       |
| Inspection Type | <b>Initial Inspection</b>                           |
| Permit          | <b>Test Permit</b>                                  |
| Scheduled Date  | * 9/25/2019 <input type="button" value="Calendar"/> |
| Owner           | * <b>MOD Administrator</b>                          |

- Click **New** again.
- Enter **Mechanical Inspection** for **Name**, select **Initial Inspection** for **Type**, select a permit, provide **Schedule Date**, select **New Request** for **Status Reason**, and click **Save**.

4. Lock Permit.

- Select **Inspections**.
- Make sure you have four inspection records and with various Status Reason value.

## Active Inspections ▾

| Name                  | Status Reason | Inspection Type    |
|-----------------------|---------------|--------------------|
| Electric Inspections  | Failed        | Initial Inspection |
| Framing Inspection    | Passed        | Initial Inspection |
| Mechanical Inspection | New Request   | Initial Inspection |
| Plumbing Inspections  | Pending       | Initial Inspection |

- Select **Permits**.
- Click to open the **Test Permit**.

## Active Permits ▾

| Name        |
|-------------|
| Test Permit |

- Make sure the Status Reason is set to Active and click **Lock Permit**.

|             |                  |
|-------------|------------------|
| Name        | * Test Permit    |
| Permit Type | New Construction |
| Build Site  | [Empty]          |

- The Custom Action should run. Click **Refresh**.

The screenshot shows the 'Test Permit' page. At the top, there is a toolbar with buttons for 'New', 'Deactivate', 'Lock Permit', 'Delete', 'Refresh' (highlighted with a red arrow), 'Assign', 'Share', 'Email a Link', and more. Below the toolbar, the page title 'Test Permit' and subtitle 'Permit' are displayed. On the right, a dropdown menu titled 'Active Status Reason' is open, showing 'Locked' as the selected option.

- The **Status Reason** value should change to **Locked**

The screenshot shows the 'Test Permit' page again. The 'Status Reason' dropdown is now highlighted with a red box and shows 'Locked' as the selected value. The rest of the page content remains the same, including the general information and tabs.

5. Check if the Pending and New Request Inspections get canceled

- Select Inspections.
- You should now have two canceled inspections.

The screenshot shows the 'Test Permit' page with the 'Inspections' tab selected. A timeline view shows two inspection items: 'Electric Inspections' (red box) and 'Framing Inspections' (green box). Both items are listed under the date 'Thu 5 December'. The timeline scale includes 00:00, 04:00, 08:00, 12:00, and 16:00.

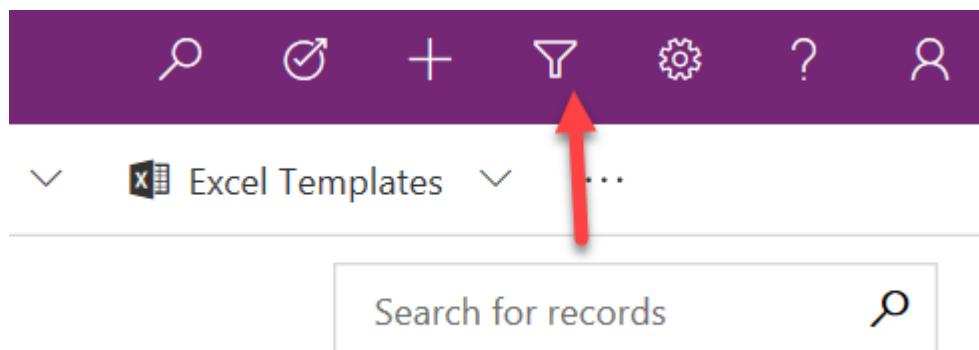
If you completed the PCF module:

The screenshot shows a sidebar navigation menu on the left with options like Home, Recent, Pinned, Permits, Inspections (selected), and Permit Types. To the right, a list of 'Active Inspections' is shown, including 'Electric Inspections' (Failed), 'Framing Inspection' (Passed), 'Mechanical Inspection' (Cancelled), and 'Plumbing Inspections' (Cancelled).

If you did not complete the PCF module:

6. Check if the Note record was created.

- Click **Advanced Find**.



- Select Notes and click Results.

The screenshot shows the 'ADVANCED FIND' tab selected in the ribbon. Below it is a toolbar with icons for Query, Saved Views, Results, New, Save, and View. To the right is a panel with Group AND, Group OR, Details, and Query options. At the bottom is a search bar labeled 'Look for:' containing 'Notes', which is highlighted with a red box. A red arrow points to the 'New' button in the toolbar.

- You should at least one Note record. Click to open the Note record.

The screenshot shows a list view of notes. At the top are buttons for Show, View, Query, and Debug. Below is a toolbar with icons for Run Workflow..., Start Dialog, and More Actions. The main area displays a table with columns for Title (with an up arrow), Description, and File Name. A red arrow points to the 'Permit Locked' column header. The table rows show entries like 'Permit Locked' and 'Reason for locking this permit:...'.

- The Regarding Column should be set to the Permit you locked.

 **Note: Permit Locked**

Note

Title \*  X

Reason for locking this permit: Admin Lock

Regarding  

 **File Attachment**

File Name:

- Close the **Note** record. Close **Advanced Find**.

### 31.2 Task #2: Test Restrict New Permit Creation Plugin

1. Try to create new Permit record for the One Microsoft Way Build Site

- Select **Permits**.
- Click **New**.

 Show Chart  New  Delete |  Refresh  Email a Link

**Active Permits** 

| Name        | Build Site |
|-------------|------------|
| Test Permit | One Mic    |

- Provide the information below and click **Save**.

Save  Save & Close  New  Lock Permit  Flow

**PERMIT**  
New Permit

Status Reason  
Active

**General** Inspections

|             |                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------|
| Name        | * Test Two Permit                                                                                   |
| Permit Type |  New Construction  |
| Build Site  |  One Microsoft Way |
| Contact     |  John Doe          |
| Start Date  | * 9/26/2019                                                                                         |
| New Size    | * 4,000                                                                                             |

- You should get the error below. Click **OK**.

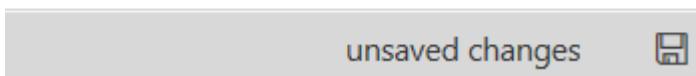


Too many locked permits for build site

Show Details

OK

- The record should not get created.



- Select **Permits**.
- Click **Discard**.

## Unsaved changes

X

Do you want to save your changes before leaving this page?

Save and continue

Discard changes

- You should have only one Permit record.

The screenshot shows a user interface for managing permits. On the left, there's a sidebar with links for Home, Recent, Pinned, and Permits. The 'Permits' link is currently selected. The main area is titled 'Active Permits' and displays a list with a single item: 'Test Permit'. There are filter and search options at the top of the list view.

## 32 Exercise #4: Plugin Trace Log and Debugging

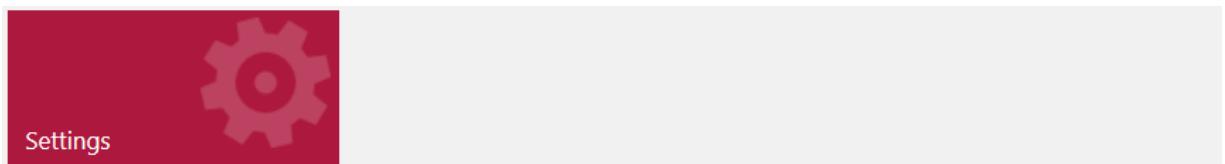
**Objective:** In this exercise, you will check the Plugin Trace log and debug the plugins

### 32.1 Task #1: Plugin Trace Log

1. Open Plugin trace Log.
  - Go back to the Permit Management application.
  - Click **Settings** and select **Advanced Settings**.

The screenshot shows the 'Settings' menu with various icons. Below the menu, there are several options: 'Link', 'Personalization Settings', 'Advanced Settings' (which has a red arrow pointing to it), and 'Toast Notification Display T...'. The 'Advanced Settings' option is the one being selected.

- Click **Settings** and select **Plugin Trace Log**.



| Business             | Customization       | System          |
|----------------------|---------------------|-----------------|
| Business Manageme... | Customizations      | Administration  |
| Templates            | Solutions           | Security        |
|                      | Microsoft AppSource | Data Management |
|                      | Plug-In Trace Log   | System Jobs     |

- You should see at least two logs.

|  | System Creat... | Operation Ty... | Type Name                       | Message Name       |
|--|-----------------|-----------------|---------------------------------|--------------------|
|  | Yes             | Plug-in         | ContosoPackageProject.PreOp...  | Create             |
|  | Yes             | Plug-in         | ContosoPackageProject.Contos... | contoso_LockPermit |

2. Open the log and see what was logged.

- Click to open the Lock Permit log.

|  | System Creat... | Operation Ty... | Type Name                       | Message Name       |
|--|-----------------|-----------------|---------------------------------|--------------------|
|  | Yes             | Plug-in         | ContosoPackageProject.PreOp...  | Create             |
|  | Yes             | Plug-in         | ContosoPackageProject.Contos... | contoso_LockPermit |

- Scroll down to the Execution section.
- Examine your Trace messages.

## Execution

| Performance          |                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Execution Start Time | 9/4/2019 2:20 PM                                                                                                                                                                                                                                                                                       |
| Message Block        | Updating Permit Id : 5c9759c4-f5c8-e911-a981-000d3a3638df, Correlation Id: 1c28df19-56d6-4dbe-bf<br>Updated Permit Id 5c9759c4-f5c8-e911-a981-000d3a3638df, Correlation Id: 1c28df19-56d6-4dbe-bf<br>Retrieving inspections for Permit Id 5c9759c4-f5c8-e911-a981-000d3a3638df, Correlation Id: 1c28df |

### 32.2 Task #2: Debugging Plugins (Optional)

32.3 Follow these steps to debug your plugins <https://docs.microsoft.com/en-us/powerapps/developer/common-data-service/tutorial-debug-plug-in>

### 32.4 lab: title: 'Lab 07: Azure Functions'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

### 32.5 Lab 07 – Azure Functions

## 33 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will create an Azure Function that will handle routing inspections. Every morning people call in to request inspections on their permits. They must call before 9:30 AM and once that period ends all the inspections for the day must be assigned and sequenced for the inspectors. To accomplish this, you will build an Azure Function that runs on a schedule, queries pending inspections and assigns them to inspectors. Given the limited time to complete the lab, we've simplified the routing and sequencing decisions.

## 34 High-level lab steps

As part of building the Azure Function, you will complete the following:

- Configure an application user for the app along with a security role
- Build the function logic to route the requests

Deploy the Azure Function

### 34.1 Things to consider before you begin

- Could we have used Dynamics 365 Universal Resource Scheduling instead of custom code?
- Could we have used Power Automate instead of custom code?
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production soon.

## 35 Exercise #1: Configure an application user

**Objective:** In this exercise, you will configure an application user that will be used to connect the Azure Function back to the Microsoft Dataverse.

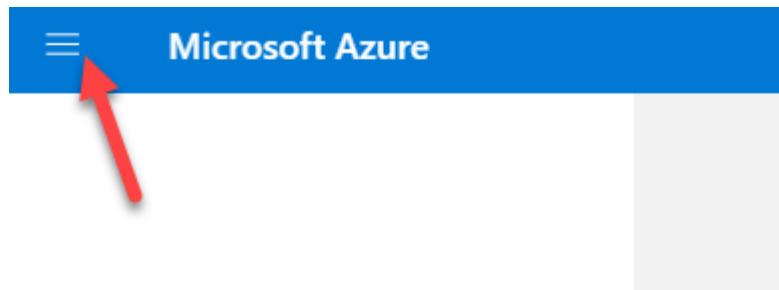
### 35.1 Task #1: Register Azure AD Application

1. Navigate to Azure Active Directory

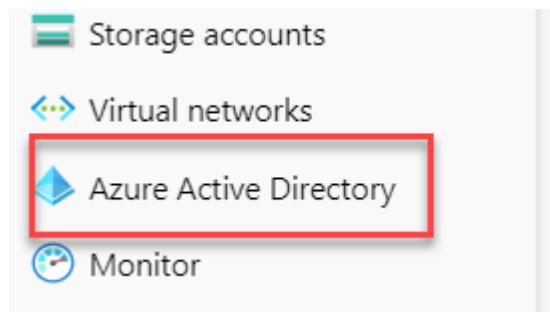
- Sign in to [Azure Portal](#).

**Note:** You must be logged in with an organization account in the same tenant as your Microsoft Dataverse Environment. This does **NOT** have to be the account that has your Azure subscription.

- Click Show portal menu.



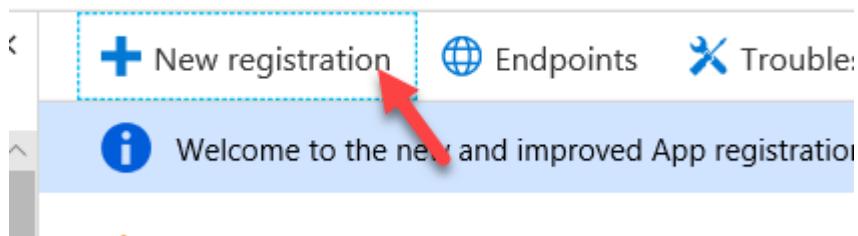
- Select **Azure Active Directory**.



- Select **App Registration** from the **Manage** section.

2. Create new application registration.

- Click **+ New registration**.



- Enter **Inspection Router** for **Name**, select **Accounts in this Organizational Directory Only** for **Supported Account Types**, and click **Register**.

### \* Name

The user-facing display name for this application (this can be changed later).

Inspection Router



### Supported account types

Who can use this application or access this API?

- Accounts in this organizational directory only (Contoso only - Single tenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

[Help me choose...](#)

### Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Web  e.g. <https://myapp.com/auth>

[By proceeding, you agree to the Microsoft Platform Policies](#)

[Register](#)

### 3. Add Client Secret.

**Note:** In this lab we are using a secret, however, you can also use a certificate. With either of these options you need to have a plan in place to handle rollover when they expire to ensure that the app keeps running.

- Select Certificates & Secrets.

Manage

Branding

Authentication

Certificates & secrets

API permissions

|             |         |
|-------------|---------|
| Application | b1513b1 |
| Director    | 3b8a00c |
| Object I    | d0cf34e |

- Click + New Client Secret.

Manage

- Branding
- Authentication
- Certificates & secrets
- Token configuration
- API permissions
- Expose an API
- Owners
- Roles and administrators (Preview)
- Manifest

Certificates can be used as secrets to prove the application's identity when requesting a token.

Upload certificate

Thumbprint

Start date

No certificates have been added for this application.

Client secrets

A secret string that the application uses to prove its identity when requesting a token.

+ New client secret

| Description        | Expires |
|--------------------|---------|
| Inspection Routing |         |

Description

Expires

In 1 year

In 2 years

Never

Add Cancel

- Enter **Inspection Routing** for Description and click Add.

Inspection Routing

Value

Copy

Cancel

- Copy the Value and save it on a notepad. You need this value in future tasks.

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

| DESCRIPTION        | EXPIRES  | VALUE                            |
|--------------------|----------|----------------------------------|
| Inspection Routing | 9/5/2020 | 8qYOPEWy4lNu--bd[G/KY6aVp0gXQQx3 |

## 35.2 Task #2: Create Application User and Security Role

In this task, you will create the application user and associate it with the Azure AD app that you just registered. You will also create a security role needed to run the routing logic.

1. Navigate to security settings of your Dev environment
- Sign in to Power Platform admin center
- Select **Environments**.
- Select the **Dev** environment, click on the ... **More Environment Actions** button and select **Settings**.

| Environment  | Type       | State |
|--------------|------------|-------|
| Rage Dev     | Production | Ready |
| Rage Prod    | Production | Ready |
| Contoso Test | Production | Ready |

- Expand **Users + Permissions**.

▽ **Product**

Behavior, Features, Languages, Privacy + Security

▽ **Business**

Business closures, Calendar, Connection roles, Currencies

▽ **Users + permissions**

Business units, Hierarchy security, Mobile configuration, Positions

▽ **Audit and logs**

- Select **Security Roles**.

Business closures, Calendar, Connection roles, Currencies

↖ **Users + permissions**

Business units

Hierarchy security

Mobile configuration

Positions

Security roles

Teams

2. Create New Security Role

- Click **New role**.

[New role](#)

[Go to legacy](#)

## Environments > Rage Dev > Settings > Security roles

Manage security roles within this environment so that people can access their data. [Learn more](#)

### Business unit

ragedev

- Enter **Inspection Router** for **Role Name** and click **Save**.
- Select the **Business Management** tab.

### Security Role: Inspection Router

Details Core Records Service Business Management Customization Missing Entities

Role Name \*

#### When role is assigned to a team

Team member gets all team privileges by default.

Team members can inherit team privileges directly based on access level. [Learn More](#)

- Locate the **User** Table and set **Read** and **Append To** privileges to **Organization**.

### Security Role: Inspection Router

| Entity                         | Create                | Read                             | Write                 | Delete                | Append                | Append To             | Assign                           | Share                 |
|--------------------------------|-----------------------|----------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|----------------------------------|-----------------------|
| Business Unit                  | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Sync Attribute Mapping Profile | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Team                           | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| User                           | <input type="radio"/> | <input checked="" type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/> |

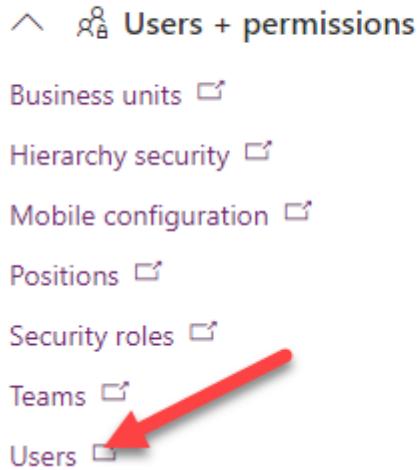
- Select the **Custom Tables** tab.
- Locate the **Inspection** Table and set **Read**, **Write**, **Append**, and **Assign** privileges to **Organization**.

| Entity                      | Create                | Read                             | Write                            | Delete                | Append                           | Append To             | Assign                           | Share                 |
|-----------------------------|-----------------------|----------------------------------|----------------------------------|-----------------------|----------------------------------|-----------------------|----------------------------------|-----------------------|
| AI Form Processing Document | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| AI Model                    | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Environment Variable Value  | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Inspection                  | <input type="radio"/> | <input checked="" type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/> | <input checked="" type="radio"/> | <input type="radio"/> |
| Knowledge Article Template  | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> | <input type="radio"/>            | <input type="radio"/> |

- Click **Save and Close**.
  - Close the **Security Roles** browser tab/window.
  - Click on the browser back button.
3. Navigate to the Application User form

- Expand **Users + permissions** again.
- Click **Users**.

Business closures, Calendar, Connection roles, Currencies



- Click **Manage users in Dynamics 365**.



Environments > Rage Dev > Settings > **Users**

- Switch to the **Application Users** view.

The screenshot shows the 'Enabled Users' view with the following interface elements:

- Top navigation: NEW, PROMOTE TO ADMIN, EMAIL A LINK, FLOW, FLOW, RUN REPORT, X
- Left sidebar: Enabled Users, System Views (Administrative Access Users, Application Users, Associated Record Team Members, Disabled Users), and a list of users.
- Right pane: A table showing user details.

| Business Unit | Title    | Position |
|---------------|----------|----------|
| orga5253b09   | IT Admin |          |

- Click **+ New**.

A screenshot of a Microsoft PowerApp interface. At the top, there are several buttons: '+ NEW' (with a red arrow pointing to it), 'PROMOTE TO ADMIN', 'EMAIL A LINK', 'FLOW', 'FLOW', and 'RUN RE'. Below this is a header bar with a 'User' icon and the text 'Application Users'. A red box highlights this header. The main area shows a table with columns: 'Full Name ↑', 'Application I...', and 'Azure AD Obj...'. One row is visible: 'PowerApps Checker Application' with ID 'c9299480-c1...' and object ID '4f2aeba9-420...'. At the bottom are buttons for 'SAVE', 'SAVE & CLOSE', 'FLOW', 'FLOW', and 'FORM EDITOR'.

- Switch to the **Application User** form.

A screenshot of the 'Application User' form. At the top left is a user icon and a 'USER' dropdown menu. The menu has three options: 'User', 'Application User' (which has a red arrow pointing to it), and 'User form – Business'. Below the menu is a yellow status bar with the message: 'The information provided in this form is viewable by the entire organization.' At the bottom left is a 'Summary' button.

4. Create Application User.

- Click on the **Full Name** Column.
- Enter [InspectionRouter@Tenant.onmicrosoft.com](mailto:InspectionRouter@Tenant.onmicrosoft.com) for User name, **Inspection** for First Name, **Router** for Last Name, enter [InspectionRouter@Tenant.onmicrosoft.com](mailto:InspectionRouter@Tenant.onmicrosoft.com) for Email.



## New User

User · Application User ▾

### Summary

#### Account Information

User Name \* InspectionRouter@M365x920496.onmicrosoft.com

Application ID \* ---

Application ID URI ---

Azure AD Object ID \* ---

#### User Information

First Name \* Inspection

Last Name \* Router

Primary Email \* InspectionRouter@M365x920496.onmicrosoft.com

#### 5. Copy App ID from Azure.

- Go back to your **Azure** portal.
- Select **Azure Active Directory**.
- Select **App Registrations | Owned applications**
- Click to open the registration you created.

The screenshot shows the Azure portal's left-hand navigation menu and the 'Owned applications' section of the 'App registrations' page.

**Left Navigation:**

- Resource groups
- App Services
- Function App
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Azure Active Directory** (highlighted with a red box)

**Owned applications:**

- All applications
- Owned applications** (selected)

**Search bar:** Start typing a name or Application ID to filter these results

**DISPLAY NAME:**

|    |                   |
|----|-------------------|
| IR | Inspection Router |
|----|-------------------|

A red arrow points from the 'App registrations' link in the navigation menu to the 'Inspection Router' entry in the list.

- Copy the **Application (Client ID)**.

 Delete  Endpoints

 Welcome to the new and improved App registrations. Looking to learn how it's changed from App Registrations? [Get started](#)

Display name : Inspection Router

[Copy to clipboard](#)

Application (client) ID : b1513bf1-5d08-43d3-8507-48a646536d28



Directory (tenant) ID : 3b8a00d9-abe1-4b3a-8bc1-f57c73196cb9

Object ID : d0cf34ea-5c81-471f-ba5f-a67c3d9f39ae

6. Complete the Application User creation

- Paste the Application Id you copied.

 **New User**  
User · Application User ▾

## Summary

### Account Information

User Name \* **InspectionRouter@M365x920496.onmicrosoft.com**

Application ID \* **596...3c30fa**

- Click **Save**.
- The **Application ID URI** and **Azure AD Object ID** Columns should auto populate.

 **Inspection Router**  
User · Application User ▾

## Summary

### Account Information

User Name \* **InspectionRouter@M365x920496.onmicrosoft.com**

Application ID \* **596...3c30fa**

 Application ID URI **596...0fa**

 Azure AD Object ID \* **bcl...0576**

- Close the **User** form.

7. Assign the Inspection Router security role to the Application User you created
  - Refresh the user view, select the user you created, and click **Manage Roles**.

The screenshot shows a Dynamics 365 interface for managing application users. At the top, there are several buttons: NEW, EDIT, APPROVE EMAIL, REJECT EMAIL, PROMOTE TO ADMIN, MANAGE ROLES (which has a red arrow pointing to it), CHANGE BUSINESS UNIT, and CHANGE MANAGER. Below this is a header bar with 'Application Users' and a dropdown arrow. Underneath is a table with columns for 'Full Name ↑', 'Application I...', 'Azure AD Obj...', and 'Application I...'. A single row is selected, showing 'Inspection Router' in the first column and three GUIDs in the other columns. The 'MANAGE ROLES' button is highlighted with a red arrow.

| Full Name ↑                                | Application I... | Azure AD Obj... | Application I... |
|--------------------------------------------|------------------|-----------------|------------------|
| <input type="checkbox"/> Inspection Router | 43936f75-247...  | 2795414c-51...  | 43936f75-247...  |

- Select **Inspection Router** and click **OK**.

|                                                       |             |
|-------------------------------------------------------|-------------|
| <input type="checkbox"/> Environment Maker            | orga5253b09 |
| <input type="checkbox"/> Export Customizations        | orga5253b09 |
| <input checked="" type="checkbox"/> Inspection Router | orga5253b09 |
| <input type="checkbox"/> Knowledge Manager            | orga5253b09 |
| <input type="checkbox"/> Solution Checker             | orga5253b09 |

**i** As you assign security roles to your users, you will enable access and the ability to extract your data. Access is enabled through multiple clients (i.e. Dynamics 365 for Outlook, Dynamics 365 for tablets, web-user). You may administer these access privileges by

OK

Cancel

## 36 Exercise #2: Create Azure Function for Inspection Routing

**Objective:** In this exercise, you will create the Azure function that will route the inspections.

### 36.1 Task #1: Create the Function

- Create Azure Function project
  - Start **Visual Studio**.
  - Click **Create a New Project**.

Get started

Open a local Visual Studio project or .sln file

 Open a local folder  
Navigate and edit code within any folder

 Create a new project   
Choose a project template with code scaffolding to get started

[Continue without code →](#)

- Select **Azure Functions** and click **Next**.

C#    Android    iOS    Linux    macOS    Windows    Library

 Azure Functions  
A template to create an Azure Function project.  
[C#](#)    [Azure](#)    [Cloud](#)

 Blank App (Universal Windows)  
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.  
[C#](#)    [Windows](#)    [Xbox](#)    [UWP](#)    [Desktop](#)

[Back](#)    [Next](#)

- Enter **InspectionManagementApp** for **Name** and click **Create**.

## Configure your new project

Azure Functions C# Azure Cloud

Project name

Location

Solution

Solution name 

Place solution and project in the same directory

- Select **Azure Function V3 (.NET Core)** and select **Timer Trigger**.

## Create a new Azure Functions Application

Azure Functions v3 (.NET Core)



### Http trigger

A C# function that will be run whenever it receives an HTTP request



### Service Bus Topic trigger

A C# function that will be run whenever a message is added to the specified Service Bus topic



### Timer trigger

A C# function that will be run on a specified schedule

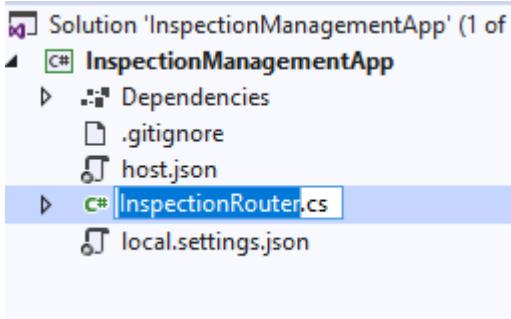
- Change the Schedule to **0 0 0 \*\*\*** (Midnight Every Day) and click **Create**.

# Create a new Azure Functions Application

The screenshot shows the Azure Functions v3 (.NET Core) creation wizard. It includes sections for selecting triggers (Http trigger, IoT Hub trigger, Timer trigger), storage account configuration (Storage Account: AzureWebJobsStorage, Storage Emulator), and scheduling (Schedule: 0 0 \* \* \*). A 'Get started with Azure Functions' link and 'Back' and 'Create' buttons are also visible.

## 2. Rename and run the Function

- Rename the function file **InspectionRouter**.

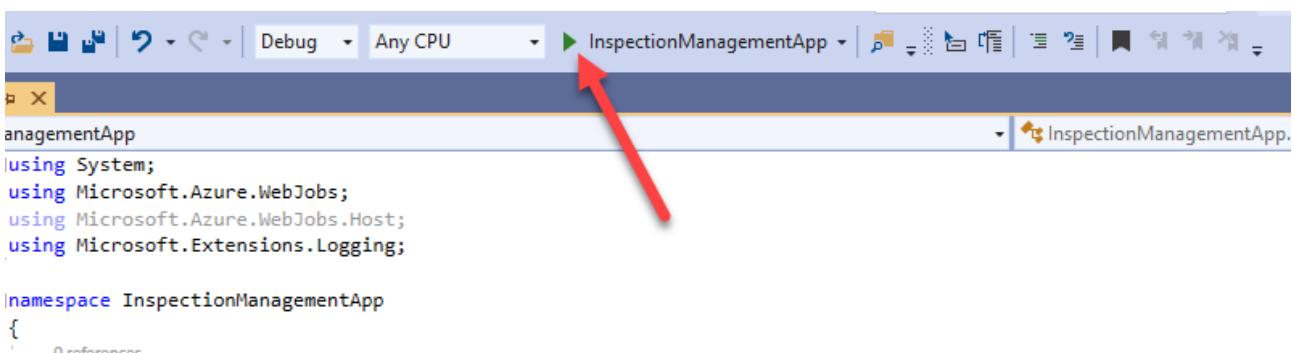


- Change the function class and FunctionName to **InspectionRouter**.

```
namespace InspectionManagementApp
{
 public static class InspectionRouter
 {
 [FunctionName("InspectionRouter")]
 public static void Run([TimerTrigger("0 0 0 * * *")]Time
 {
 log.LogInformation($"C# Timer trigger function execut

```

- Click **Run**.



- You should see **Now Listening on:** <http://0.0.0.0:7071>

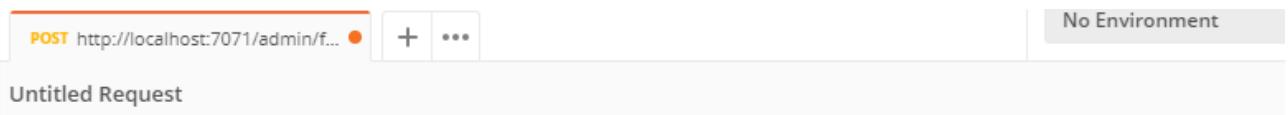
```

Hosting environment: Production
Content root path: C:\Users\hrage\Documents\Dev Labs\Day 3\Mod 2
ug\netcoreapp2.1
Now listening on: http://0.0.0.0:7071
Application started. Press Ctrl+C to shut down.
[9/5/2019 4:42:10 PM] Host lock lease acquired by instance ID '0'

```

3. Trigger the function with Postman

- If you don't have **Postman** installed, get it from here [Postman](#) and install it on your machine. Or you can use any similar tool of your preference that allows you to construct an HTTP POST request.
- Start **Postman**.
- Select **POST** and paste the URL: <http://localhost:7071/admin/functions/InspectionRouter>



Untitled Request

|              |               |                                                        |             |                    |       |        |
|--------------|---------------|--------------------------------------------------------|-------------|--------------------|-------|--------|
| POST         | ▼             | http://localhost:7071/admin/functions/InspectionRouter | <b>Send</b> |                    |       |        |
| Params       | Authorization | Headers                                                | Body        | Pre-request Script | Tests | Cookie |
| Query Params |               |                                                        |             |                    |       |        |

- Select the **Headers** tab.
- Add **Content-Type** and set it to **application/json**.

|                                                  |               |                    |             |                    |       |         |
|--------------------------------------------------|---------------|--------------------|-------------|--------------------|-------|---------|
| Params                                           | Authorization | <b>Headers (1)</b> | Body        | Pre-request Script | Tests | Cookies |
| ▼ Headers (1)                                    |               |                    |             |                    |       |         |
| KEY                                              |               | VALUE              | DESCRIPTION | ...                | Bul   |         |
| <input checked="" type="checkbox"/> Content-Type |               | application/json   |             |                    |       |         |
| Key                                              |               | Value              | Description |                    |       |         |

- Select the **Body** tab.
- Select **Raw** and set it to empty json {}.

|                            |                                 |                                                        |                                      |                              |                                           |             |  |
|----------------------------|---------------------------------|--------------------------------------------------------|--------------------------------------|------------------------------|-------------------------------------------|-------------|--|
| POST                       | ▼                               | http://localhost:7071/admin/functions/InspectionRouter |                                      |                              |                                           |             |  |
| Params                     | Authorization                   | <b>Headers (1)</b>                                     | <b>Body</b> ●                        | Pre-request Script           | Tests                                     |             |  |
| <input type="radio"/> none | <input type="radio"/> form-data | <input type="radio"/> x-www-form-urlencoded            | <input checked="" type="radio"/> raw | <input type="radio"/> binary | <input type="radio"/> GraphQL <b>BETA</b> | <b>JSON</b> |  |
| 1 {}                       |                                 |                                                        |                                      |                              |                                           |             |  |

- Click **Send**.
- You should get **202 Accepted Status**.

Status: 202 Accepted Time: 1369ms Size: 98 B | [Sa](#)

- Go to the output console.
- The function should get triggered.

```
9/5/2019 5:09:25 PM] type: "WebJobsAuthLevel" ,
9/5/2019 5:09:25 PM] "level": "Admin"
9/5/2019 5:09:25 PM] }
9/5/2019 5:09:25 PM]],
9/5/2019 5:09:25 PM] "status": 202,
9/5/2019 5:09:25 PM] "duration": 659
9/5/2019 5:09:25 PM] }
9/5/2019 5:09:25 PM] Executing 'InspectionRouter' (Reason='This function was programmatically called by host system')
, Id=848cc9cc-32d2-4e27-b3f3-3925bb7f3556)
9/5/2019 5:09:25 PM] C# Timer trigger function executed at: 9/5/2019 11:09:25 AM
9/5/2019 5:09:25 PM] Executed 'InspectionRouter' (Succeeded, Id=848cc9cc-32d2-4e27-b3f3-3925bb7f3556)
```

- Go back to **Visual Studio** and stop debugging.

#### 4. Add NuGet packages

- Right Click on the project and select **Manage NuGet Packages**.
- Select the **Browse** tab and search for **Microsoft.IdentityModel.Clients.ActiveDirectory**.
- Select the latest stable version and click **Install**.

The screenshot shows the NuGet Package Manager interface. The search bar contains 'Microsoft.IdentityModel.Clients.ActiveDirectory'. On the left, there are tabs for 'Browse', 'Installed', and 'Updates'. On the right, it says 'NuGet Package Manager: InspectionManagementApp' and 'Package source: nuget.org'. The results list shows two packages:

- Microsoft.IdentityModel.Clients.ActiveDirectory** by Microsoft, 37.3M downloads, version v5.2.1. A red box highlights the package name.
- Microsoft.IdentityModel.Tokens** by Microsoft, 46.3M downloads, version v5.5.0.

- Search for **Xrm.Tools.CrmWebAPI**. Note: This is a community library designed to work with the Microsoft Dataverse Web API. When you are building this type of extension you can use any oData V4 library you prefer. Make sure you select the one developed by DavidYack.
- Select the latest stable version and click **Install**.

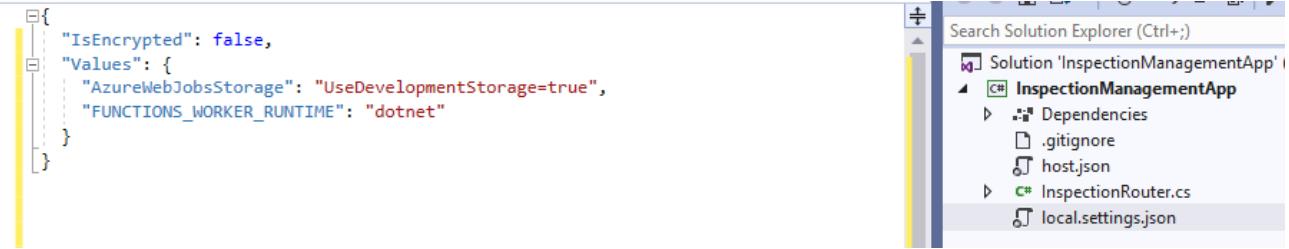
The screenshot shows the NuGet Package Manager interface. The search bar contains 'Xrm.Tools.CrmWebAPI'. On the left, there are tabs for 'Browse', 'Installed', and 'Updates'. On the right, it says 'NuGet Package Manager: InspectionManagementApp' and 'Package source: nuget.org'. The results list shows two packages:

- Xrm.Tools.CRMWebAPI** by DavidYack, 19K downloads, version v1.0.24. A red box highlights the package name.
- System.Diagnostics.Tools** by Microsoft, 58M downloads, version v4.3.0. A red box highlights the package name.

- Close the **NuGet Package Manager**.

#### 5. Edit the local settings file

- Click to open the **local.settings.json** file



- Add the **Values** below to **local.settings**

```

"cdsurl": "",

"cdsclientid": "",

"cdsclientsecret": ""

{
 "IsEncrypted": false,
 "Values": {
 "AzureWebJobsStorage": "UseDevelopmentStorage=true",
 "FUNCTIONS_WORKER_RUNTIME": "dotnet",
 "cdsurl": "",
 "cdsclientid": "",
 "cdsclientsecret": ""
 }
}

```

- Find the Client Secret you saved in the notepad and paste as the cdsclientsecret.

```

{
 "IsEncrypted": false,
 "Values": {
 "AzureWebJobsStorage": "UseDevelopmentStorage=true",
 "FUNCTIONS_WORKER_RUNTIME": "dotnet",
 "cdsurl": "",
 "cdsclientid": "",
 "cdsclientsecret": "8qYOPeW4lNu--bd[G/KY6aVp0gXQQx3"
 }
}

```

## 6. Copy the App ID

- Go back to your **Azure** portal.
- Select **Azure Active Directory**.
- Select **App Registrations**.
- Click to open the registration you created.

All applications    Owned applications

Start typing a name or Application ID to filter these results

DISPLAY NAME

IR    Inspection Router

- Resource groups
- App Services
- Function App
- SQL databases
- Azure Cosmos DB
- Virtual machines
- Load balancers
- Storage accounts
- Virtual networks
- Azure Active Directory**

**App registrations**

- Identity Governance
- Application proxy
- Licenses

- Copy the **Application (Client ID)**.

Delete    Endpoints

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations? [Get started](#)

|                         |                                        |                                   |
|-------------------------|----------------------------------------|-----------------------------------|
| Display name            | : Inspection Router                    | <a href="#">Copy to clipboard</a> |
| Application (client) ID | : b1513bf1-5d08-43d3-8507-48a646536d28 |                                   |
| Directory (tenant) ID   | : 3b8a00d9-abe1-4b3a-8bc1-f57c73196cb9 |                                   |
| Object ID               | : d0cf34ea-5c81-471f-ba5f-a67c3d9f39ae |                                   |

- Go back to **Visual Studio** and paste the **Application ID** as the `cdsclientid`.
7. Find the your Microsoft Dataverse URL
- Sign in to <https://admin.powerplatform.microsoft.com>
  - Select **Environments** and click to open the **Dev** environment.

+ New    Refresh

Environments

| Environment       | Type |
|-------------------|------|
| Rage Dev          | ...  |
| Rage Prod         | ...  |
| Contoso (default) | ...  |

Analytics

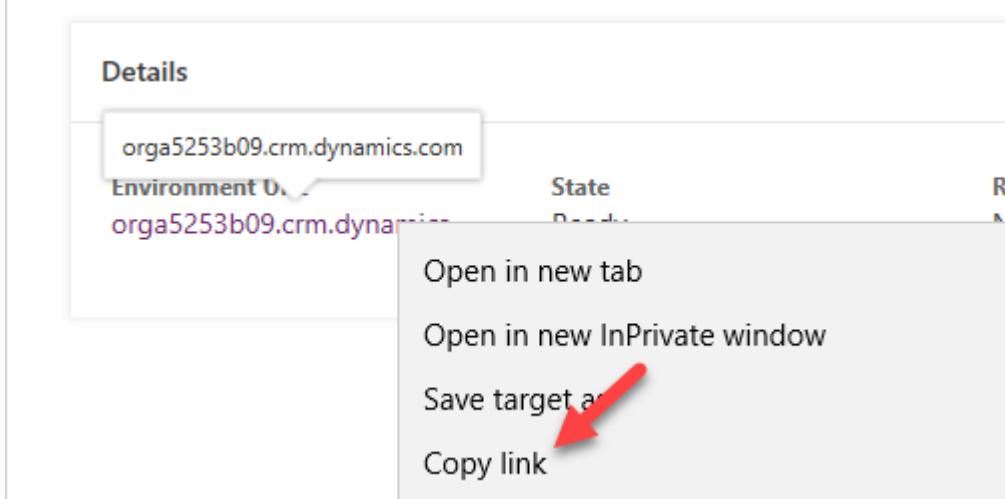
Help + support

Data integration

Data gateways

Data policies

- Copy the **Environment URL**.



- Go back to **Visual Studio** and paste the **URL** you copied as the **cdsurl**.

```

1 {
2 "IsEncrypted": false,
3 "Values": [
4 "AzureWebJobsStorage": "UseDevelopmentStorage=true",
5 "FUNCTIONS_WORKER_RUNTIME": "dotnet",
6 "cdsurl": "https://orga5253b09.crm.dynamics.com",
7 "cdsclientid": "b1513bf1-5d08-43d3-8507-48a646536d28",
8 "cdsclientsecret": "8qYOPeWwy4lNu--bd[G/KY6aVp0gXQQx3"
9]
10 }
11

```

- Save and close the file.
- 8. Add using statements to the function class.
- Open the **InspectionRouter.cs** file
- Add the using statements below.

```

using System.Threading.Tasks;
using Xrm.Tools.WebAPI;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Xrm.Tools.WebAPI.Results;
using System.Dynamic;
using Xrm.Tools.WebAPI.Requests;
using System.Collections.Generic;

```

9. Create a method that will create the web API.

- Add the method below inside the class.

```

private static async Task<CRMWebAPI> GetCRMWebAPI(ILogger log)
{
 return null;
}

```

- Add the local variables below before the return line on the **GetCRMWebAPI** method.

```

var clientID = Environment.GetEnvironmentVariable("cdsclientid", EnvironmentVariableTarget.Process);
var clientSecret = Environment.GetEnvironmentVariable("cdsclientsecret", EnvironmentVariableTarget.Process);
var crmBaseUrl = Environment.GetEnvironmentVariable("cdsurl", EnvironmentVariableTarget.Process);
var crmurl = crmBaseUrl + "/api/data/v9.0/";

```

- Create **Authentication Parameters**.

```
AuthenticationParameters ap = await AuthenticationParameters.CreateFromUrlAsync(new Uri(crmurl))
```

- Create **Client Credential** passing your **Client Id** and **Client Secret**.

```

 var clientcred = new ClientCredential(clientID, clientSecret);
• Get Authentication Context.
 // CreateFromUrlAsync returns endpoint while AuthenticationContext expects authority
 // workaround is to downgrade adal to v3.19 or to strip the tail
 var auth = ap.Authority.Replace("/oauth2/authorize", "");
 var authContext = new AuthenticationContext(auth);

• Get Token.
 var authenticationResult = await authContext.AcquireTokenAsync(crmBaseUrl, clientcred);

• Return the web API. Replace the return line with the code below.

 return new CRMWebAPI(crmurl, authenticationResult.AccessToken);

```

```

1 reference
private static async Task<CRMWebAPI> GetCRMWebAPI	ILogger log)
{
 var clientID = Environment.GetEnvironmentVariable("cdsclientid", EnvironmentVariableTarget.Process);
 var clientSecret = Environment.GetEnvironmentVariable("cdsclientsecret", EnvironmentVariableTarget.Process);
 var crmBaseUrl = Environment.GetEnvironmentVariable("cdsurl", EnvironmentVariableTarget.Process);
 var crmurl = crmBaseUrl + "/api/data/v9.0/";

 AuthenticationParameters ap = await AuthenticationParameters.CreateFromUrlAsync(new Uri(crmurl));

 var clientcred = new ClientCredential(clientID, clientSecret);

 // CreateFromUrlAsync returns endpoint while AuthenticationContext expects authority
 // workaround is to downgrade adal to v3.19 or to strip the tail
 var auth = ap.Authority.Replace("/oauth2/authorize", "");
 var authContext = new AuthenticationContext(auth);

 var authenticationResult = await authContext.AcquireTokenAsync(crmBaseUrl, clientcred);

 return new CRMWebAPI(crmurl, authenticationResult.AccessToken);
}

```

## 10. Test the web API you created

- Call the GetCRMWebAPI method. Add the code below to the Run method.

```

CRMWebAPI api = GetCRMWebAPI(log).Result;
• Execute WhoAmI function and log the User Id.
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");

```

```

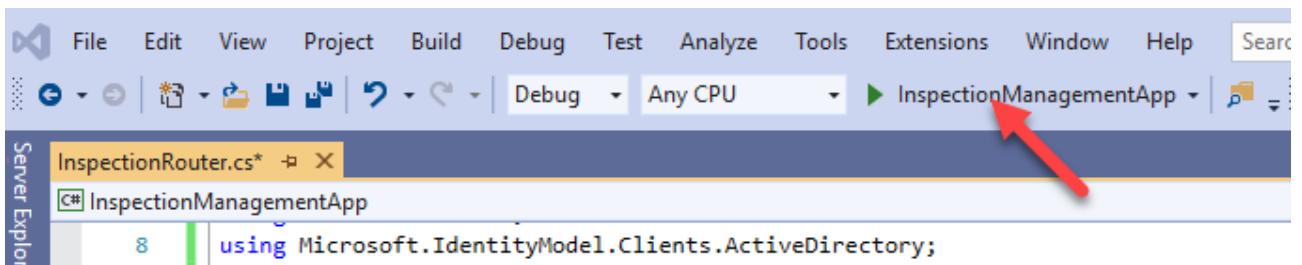
[FunctionName("InspectionRouter")]
0 references
public static void Run([TimerTrigger("0 0 0 * * * ")]TimerInfo myTimer, ILogger log)
{
 log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");

 CRMWebAPI api = GetCRMWebAPI(log).Result;
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");
}

```

## 11. Debug

- Click Run.



- Go back to **Postman** and click **Send**.

Untitled Request

POST http://localhost:7071/admin/functions/InspectionRouter **Send** Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary GraphQL BETA JSON (application/json) Beautify

```
1 { }
```

- Go to the output console.
- You should see the **User ID**.

```
., Id=edfb925f-4d2d-49cd-a26f-46dea7bc5b84)
[8/14/2020 4:40:54 AM] C# Timer trigger function executed at: 8/13/2020 10:40:54 PM
[8/14/2020 4:40:56 AM] UserID: accd59f6-e4dd-ea11-a816-00224803d22e
[8/14/2020 4:40:56 AM] Executed 'InspectionRouter' (Succeeded, Id=edfb925f-4d2d-49cd-a26f-46dea7bc5b84, Dur
```

Go back **Visual Studio** and stop debugging.

## 36.2 Task #2: Get Inspections and Users and Assign Inspections

1. Create a method that will get all active inspections that are New Request or Pending, and scheduled for today
- Add the method below inside the class.

```
private static Task<CRMGetListResult<ExpandoObject>> GetInspections(CRMWebAPI api)
{
 return null;
}
```

- Create **Fetch XML**. Add the code below before the return line of the GetInspections method.

```
var fetchxml = @"<fetch version=""1.0"" mapping=""logical"" >
<entity name=""contoso_inspection"" >
<attribute name=""contoso_inspectionid"" />
<attribute name=""contoso_name"" />
<attribute name=""ownerid"" />
<attribute name=""contoso_inspectiontype"" />
<attribute name=""contoso_sequence"" />
<attribute name=""contoso_scheduleddate"" />
<filter type=""and"" >
<condition value=""0"" operator=""eq"" attribute=""statecode"" />
<condition attribute=""contoso_scheduleddate"" operator=""today"" />
<condition attribute=""statuscode"" operator=""in"" >
<value>1</value>
<value>463270000</value>
</condition>
```

```

</filter>
</entity>
</fetch>";

```

- Get the list of Inspections.

```

var inspections = api.GetList<ExpandoObject>("contoso_inspections", QueryOptions: new CRMGetListOptions()
{
 FetchXml = fetchxml
});

```

- Return the Inspections. Replace the return line with the code below.

```

return inspections;
private static async Task<CRMWebAPI> GetCRMWebAPI	ILogger log)...

```

0 references

```

private static Task<CRMGetListResult<ExpandoObject>> GetInspections(CRMWebAPI api)
{
 var fetchxml = @"<fetch version=""1.0"" mapping=""logical"" >
 <entity name=""contoso_inspection"" >
 <attribute name=""contoso_inspectionid"" />
 <attribute name=""contoso_name"" />
 <attribute name=""ownerid"" />
 <attribute name=""contoso_inspectiontype"" />
 <attribute name=""contoso_sequence"" />
 <attribute name=""contoso_scheduleddate"" />
 <filter type=""and"" >
 <condition value=""0"" operator=""eq"" attribute=""statecode"" />
 <condition attribute=""contoso_scheduleddate"" operator=""today"" />
 <condition attribute=""statuscode"" operator=""in"" >
 <value>1</value>
 <value>463270000</value>
 </condition>
 </filter>
 </entity>
 </fetch>";

 var inspections = api.GetList<ExpandoObject>("contoso_inspections", QueryOptions: new CRMGetListOptions()
 {
 FetchXml = fetchxml
 });
}

return inspections;
}

```

- Call the GetInspections method from the Run method.

- Go back to the **Run** method.
- Call the **GetInspections** method.

```

var inspections = GetInspections(api).Result;

[FunctionName("InspectionRouter")]
0 references
public static void Run([TimerTrigger("0 0 0 * * * ")]TimerInfo myTimer, ILogger log)
{
 log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");

 CRMWebAPI api = GetCRMWebAPI(log).Result;
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");

 var inspections = GetInspections(api).Result;
}

```

- Create a method that will get all users.

- Add the method below inside the class.

```

private static Task<CRMGetListResult<ExpandoObject>> GetUsers(CRMWebAPI api)
{
 var users = api.GetList<ExpandoObject>("systemusers");
 return users;
}

```

- Call the **GetUsers** method from the **Run** method.

```

var users = GetUsers(api).Result;

[FunctionName("InspectionRouter")]
0 references
public static void Run([TimerTrigger("0 0 0 * * * ")]TimerInfo myTimer, ILogger log)
{
 log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");

 CRMWebAPI api = GetCRMWebAPI(log).Result;
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");

 var inspections = GetInspections(api).Result;
 var users = GetUsers(api).Result;
}

```

4. Create a method that will assign inspections to users

- Add the method below to the class.

```

private static async Task<CRMUpdateResult> RouteInspection(CRMWebAPI api, dynamic inspection,
{
 dynamic updateObject = new ExpandoObject();
 ((IDictionary<string, object>)updateObject).Add
 ("ownerid@odata.bind", "/systemusers(" + userId + ")");
 updateObject.contoso_sequence = sequenceNumber.ToString();
 return await api.Update("contoso_inspections", new Guid(inspection.contoso_inspectionid), updateObject);
}

```

5. Create two-digit random number.

- Add the code below to the Run method.

```

Random rnd = new Random();
int sequenceNumber = rnd.Next(10, 99);

```

6. Assign Inspections

- Go through the **Inspections** and call the **RouteInspection** method.

```

int currentUserIndex = 0;
foreach (dynamic inspection in inspections.List)
{
 log.LogInformation($"Routing inspection {inspection.contoso_name}");
 var inspectionResult = new CRMUpdateResult();
 //Your record assignment would like this. We will not assign records to different users in this lab.
 // if (users.List.Count > (currentUserIndex))
 //{
 // dynamic currentUser = users.List[currentUserIndex];
 // inspectionResult = RouteInspection(api, inspection, currentUser.systemuserid.ToString(), sequenceNumber);
 // currentUserIndex++;
 //}
}

```

- We will not assign inspection records to other users in this lab. **Comment** out the **if** statement you just added, and we will be replacing it with logic to do the routing to our user only.

```

public static void Run([TimerTrigger("0 0 0 * * * ")]TimerInfo myTimer, ILogger log)
{
 log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");

 CRMWebAPI api = GetCRMWebAPI(log).Result;
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");

 var inspections = GetInspections(api).Result;
 var users = GetUsers(api).Result;

 Random rnd = new Random();
 int sequenceNumber = rnd.Next(10, 99);

 int currentUserIndex = 0;
 foreach (dynamic inspection in inspections.List)
 {
 log.LogInformation($"Routing inspection {inspection.contoso_name}");
 var inspectionResult = new CRMUpdateResult();
 //Your record assignment would like this. We will not assign records to different users in this lab
 //if (users.List.Count > (currentUserIndex))
 //{
 // dynamic currentUser = users.List[currentUserIndex];
 // inspectionResult = RouteInspection(api, inspection, currentUser.systemuserid.ToString(), sequenceNumber).Result;
 // currentUserIndex++;
 //}
 }
}

```

- Assign inspections to the Inspection Router. Add the code below inside **foreach**.

```
//We will instead assign inspections to the user you are currently logged in as
inspectionResult = RouteInspection(api, inspection, whoami.UserId.ToString(), sequenceNumber).Result;
```

```

0 references
public static void Run([TimerTrigger("0 0 0 * * * ")]TimerInfo myTimer, ILogger log)
{
 log.LogInformation($"C# Timer trigger function executed at: {DateTime.Now}");

 CRMWebAPI api = GetCRMWebAPI(log).Result;
 dynamic whoami = api.ExecuteFunction("WhoAmI").Result;
 log.LogInformation($"UserID: {whoami.UserId}");

 var inspections = GetInspections(api).Result;
 var users = GetUsers(api).Result;

 Random rnd = new Random();
 int sequenceNumber = rnd.Next(10, 99);

 int currentUserIndex = 0;
 foreach (dynamic inspection in inspections.List)
 {
 log.LogInformation($"Routing inspection {inspection.contoso_name}");
 var inspectionResult = new CRMUpdateResult();
 //Your record assignment would like this. We will not assign records to different users in this lab
 //if (users.List.Count > (currentUserIndex))
 //{
 // dynamic currentUser = users.List[currentUserIndex];
 // inspectionResult = RouteInspection(api, inspection, currentUser.systemuserid.ToString(), sequenceNumber).Result;
 // currentUserIndex++;
 //}

 //We will instead assign inspections to the user you are currently logged in as
 inspectionResult = RouteInspection(api, inspection, whoami.UserId.ToString(), sequenceNumber).Result;
 }
}

```

Build the project and make sure that the build succeeds.

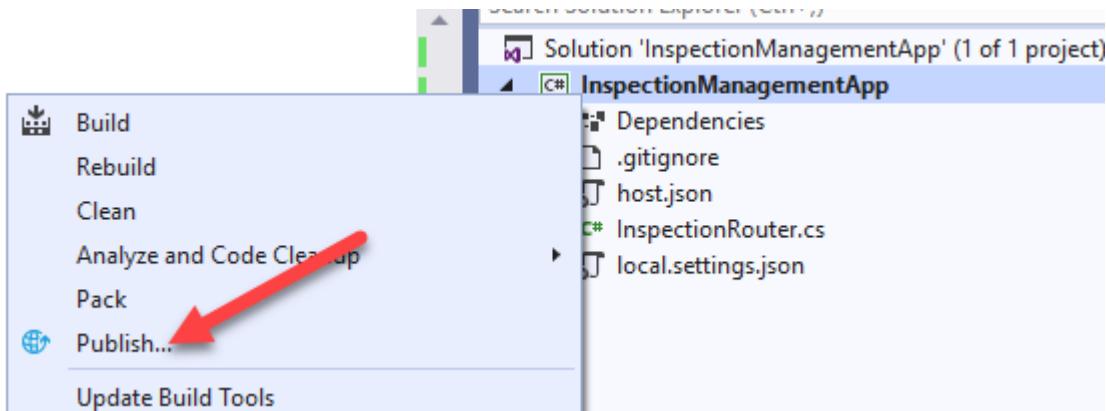
## 37 Exercise #3: Publish and Test

**Objective:** In this exercise, you will publish the Azure function to Azure, update the app settings, and test the function.

### 37.1 Task #1: Publish to Azure

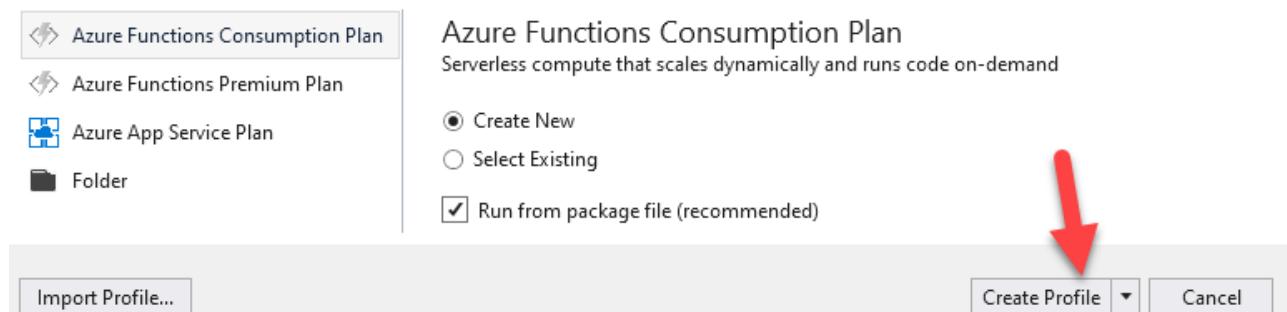
1. Publish the function

- Right click on the project and select **Publish**.

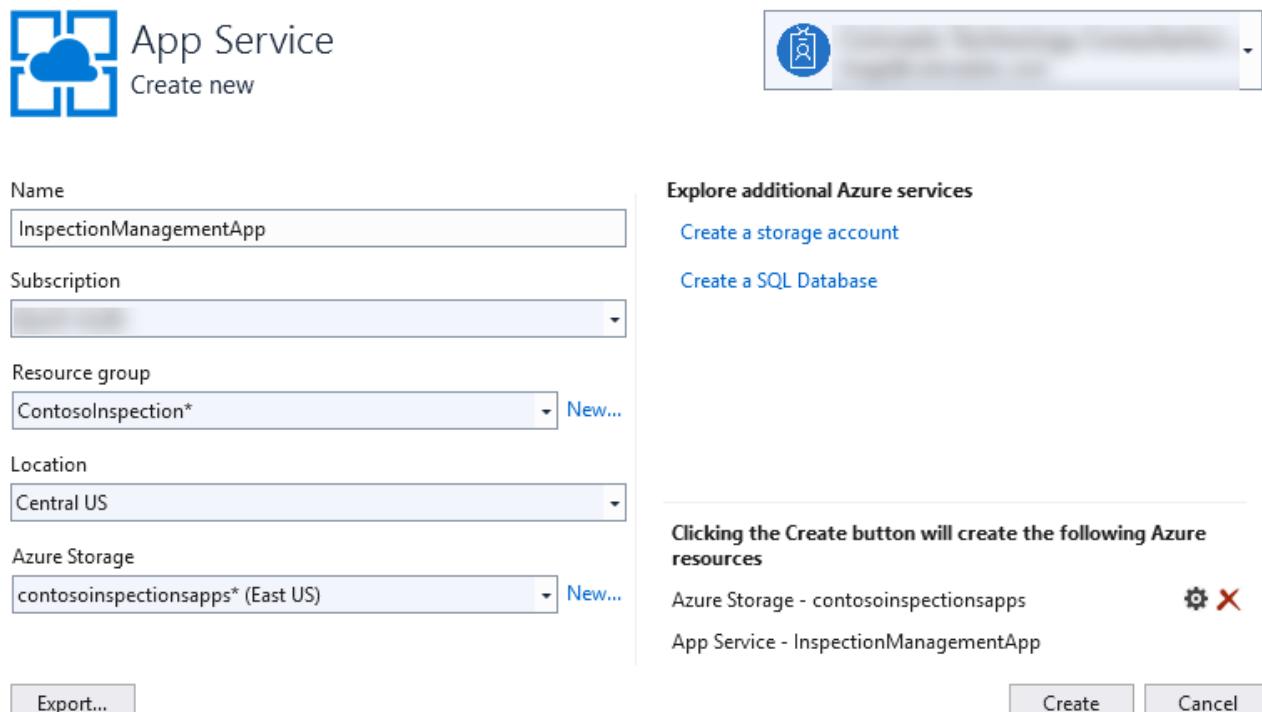


- Select **Create New** and click **Create Profile**.

#### Pick a publish target



- Make sure you are logged in to your correct **Azure** account, enter a unique name for **App Name**, create **New Resource Group**, create **New Azure Storage**, and click **Create**.

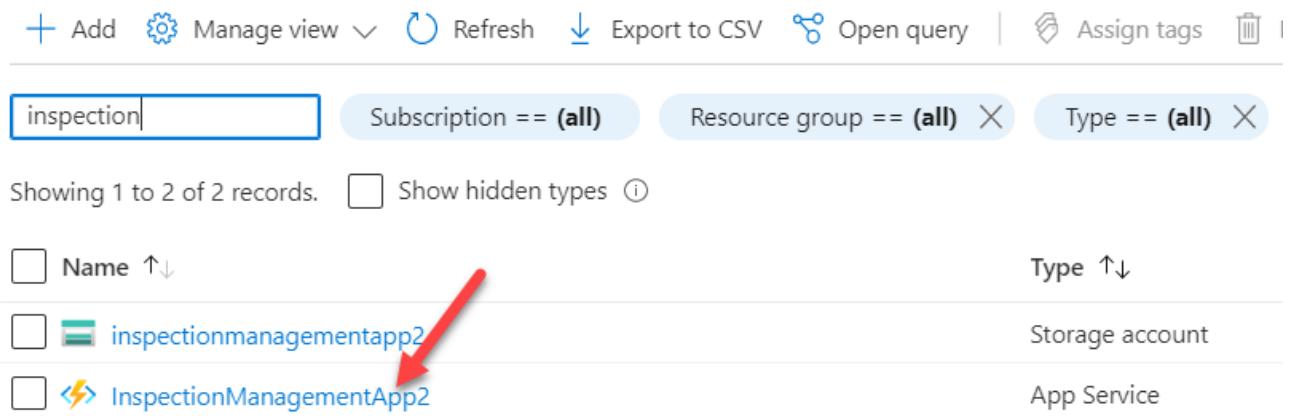


- Click **Publish**.
  - Wait for the function application to be configured and published.
2. Open the function application settings
- Go back to your **Azure** portal.
  - Select **All Resources**, search for **InspectionManagement**, and click to open the function you published.

[Home >](#)

## All resources

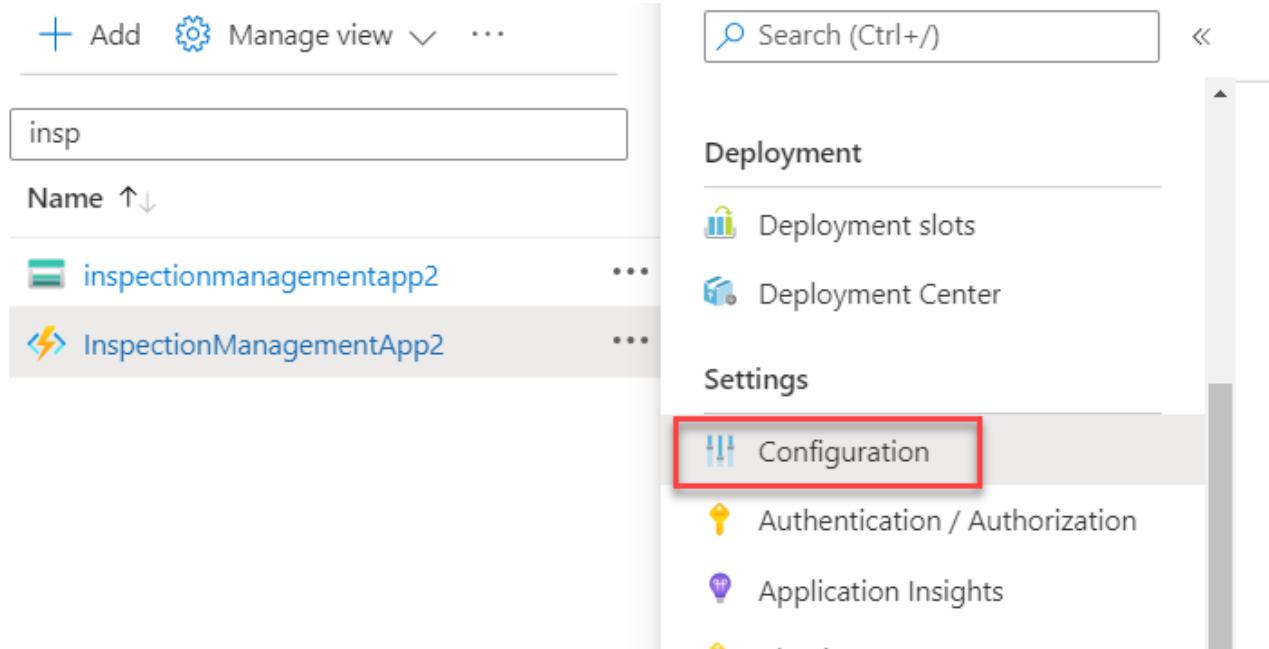
Colorado Technology Consultants, Inc.



The screenshot shows the Azure 'All resources' page with a search bar containing 'inspection'. Below the search bar, there are filters for 'Subscription == (all)', 'Resource group == (all)', and 'Type == (all)'. The results section shows two records: 'inspectionmanagementapp2' (Storage account) and 'InspectionManagementApp2' (App Service). A red arrow points to the 'InspectionManagementApp2' entry.

| Name                     | Type            |
|--------------------------|-----------------|
| inspectionmanagementapp2 | Storage account |
| InspectionManagementApp2 | App Service     |

- Click scroll down to **Settings** and select **Configuration**.



The screenshot shows the 'Settings' blade for a function app named 'InspectionManagementApp2'. The 'Configuration' tab is highlighted with a red box. Other tabs include 'Deployment slots', 'Deployment Center', 'Authentication / Authorization', 'Application Insights', and 'Identity'.

3. Update App Settings

- Click **Advanced Edit**.

 Application settings are encrypted at rest and transmitted over an encrypted channel. Your application at runtime. [Learn more](#)

 New application setting  Show values  Advanced edit  Filter

| Name                  | Value                                                                                                                |
|-----------------------|----------------------------------------------------------------------------------------------------------------------|
| AzureWebJobsDashboard |  Hidden value. Click here to view |

- Paste the json below at the top of the settings.

```
{
 "name": "cdsclientid",
 "value": "[clientid]",
 "slotSetting": false
},
{
 "name": "cdsclientsecret",
 "value": "[clientsecret]",
 "slotSetting": false
},
{
 "name": "cdsurl",
 "value": "[cdsurl]",
 "slotSetting": false
},
```

```

1 [
2 {
3 "name": "cdsclientid",
4 "value": "[clientid]",
5 "slotSetting": false
6 },
7 {
8 "name": "cdsclientsecret",
9 "value": "[clientsecret]",
10 "slotSetting": false
11 },
12 {
13 "name": "cdsurl",
14 "value": "[cdsurl]",
15 "slotSetting": false
16 },
17 {
18 "name": "AzureWebJobsDashboard",
19 "value": "DefaultEndpointsProtocol=https;AccountName=contosoinspectionsapps;A",
20 "slotSetting": false
21 },
22 {
23 "name": "AzureWebJobsStorage",

```

- Go back to **Visual Studio** and open the **local.settings.json** file.
- You will copy the **cdsurl**, **cdsclientid**, and **cdsclientsecret**. Copy the **cdsurl** value.

```

{
 "IsEncrypted": false,
 "Values": {
 "AzureWebJobsStorage": "UseDevelopmentStorage=true",
 "FUNCTIONS_WORKER_RUNTIME": "dotnet",
 "cdsurl": "https://orga5253b09.crm.dynamics.com",
 "cdsclientid": "b1513bf1-5d08-43d3-8507-48a646536d28",
 "cdsclientsecret": "8qYOPeWwy4lNu--bd[G/KY6aVp0gXQQx3"
 }
}

```

- Go back to **Azure** and replace **[cdsurl]** with the URL you copied.

```
[
 {
 "name": "cdsclientid",
 "value": "[clientid]",
 "slotSetting": false
 },
 {
 "name": "cdsclientsecret",
 "value": "[clientsecret]",
 "slotSetting": false
 },
 {
 "name": "cdsurl",
 "value": "https://orga5253b09.crm.dynamics.com",
 "slotSetting": false
 },
 {
 "name": "AzureWebJobsDashboard",
 }
]
```

- Copy the `cdsclientid` and `cdsclientsecret` values from the `local.settings.json` file and replace `[cdsclientid]` and `[cdsclientsecret]`.

```
[
 {
 "name": "cdsclientid",
 "value": "b1513bf1-5d08-43d3-8507-48a646536d28",
 "slotSetting": false
 },
 {
 "name": "cdsclientsecret",
 "value": "8qYOPEWy4lNu--bd[G/KY6aVp0gXQQx3",
 "slotSetting": false
 },
 {
 "name": "cdsurl",
 "value": "https://orga5253b09.crm.dynamics.com",
 "slotSetting": false
 },
 {
 "name": "AzureWebJobsDashboard".
 }
]
```

- Click **OK**.
- Click **Save**.

Refresh Save Discard

Application settings \* Function runtime settings General settings

## Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel.

- Click **Continue**.
- Select **Functions** and click to open the function you published.

Diagnose and solve problems

Security

Events (preview)

### Functions

Functions

App keys

- Select **Code + Test**.

Overview

Filter by name...

Name ↑↓

InspectionRouter

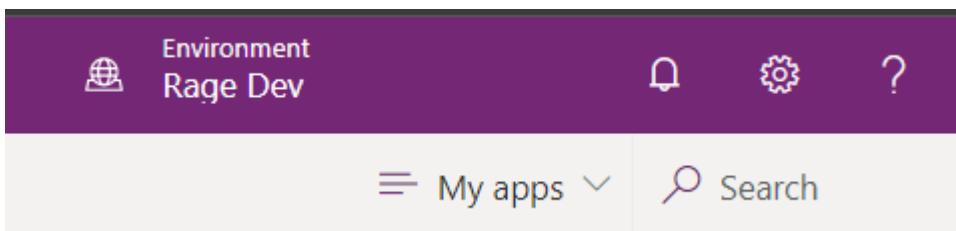
### Developer

Code + Test

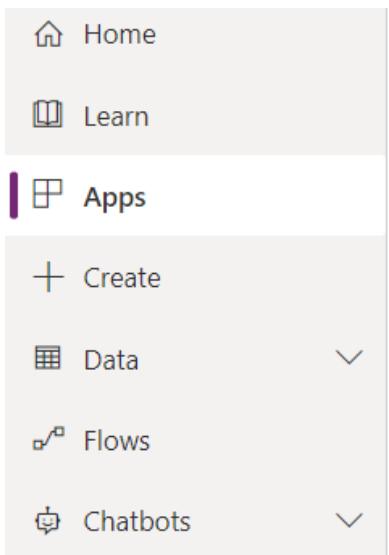
Integration

4. Prepare test record

- Start a new browser window and sign in to [Power Apps maker portal](#)
- Make sure you are in the **Dev** environment.



- Select **Apps** and click to open the **Permit Management** application.



## Apps

Apps Component libraries (preview)

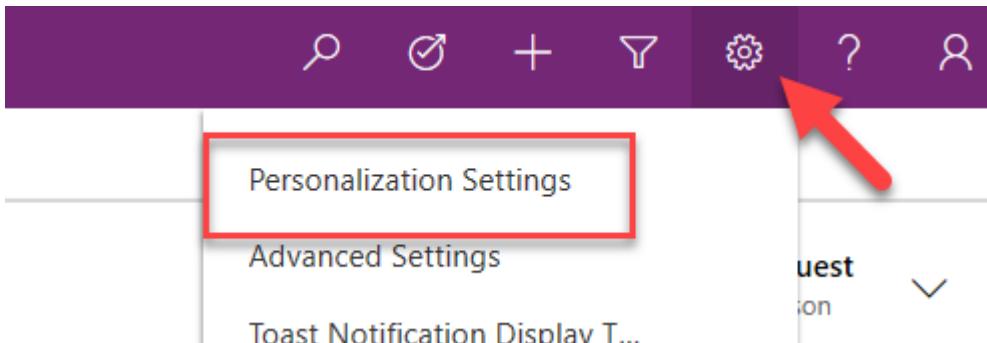
Name

Inspector

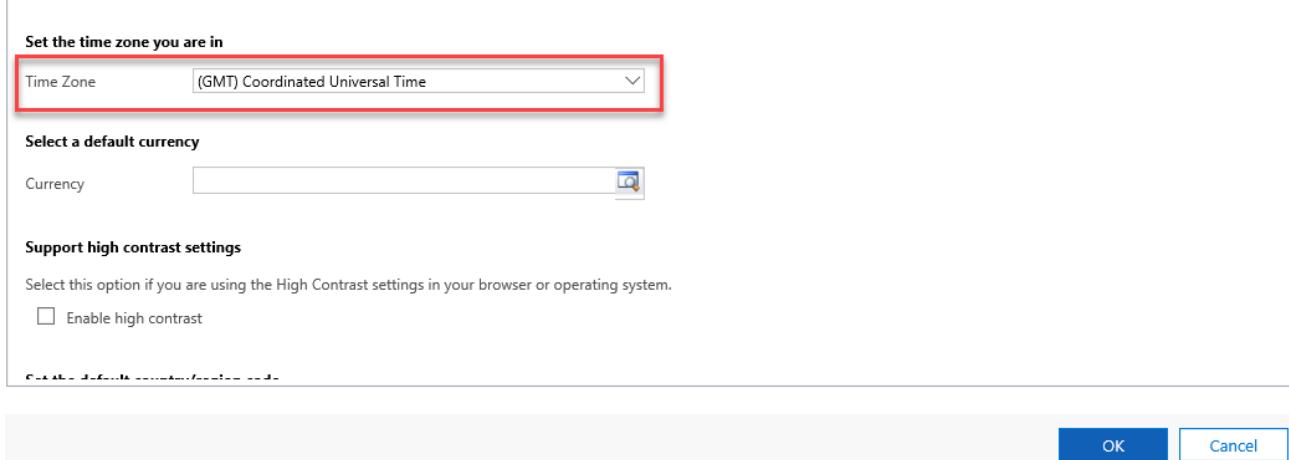
Permit Management

Solution Health Hub

- Click Settings and select **Personalization** and Settings.



- Change the **Time Zone** to **(GMT-11:00) Coordinated Universal Time-11** and click **OK**. This will ensure the query results will produce the same results regardless of your time zone.



- Select **Inspections** and click to open one of the **Inspection** records or create a new record.

The screenshot shows the 'Active Inspections' list in the Microsoft Dynamics 365 application. On the left, there is a sidebar with sections for 'Recent', 'Pinned', 'Permits', 'Permits', and 'Inspections'. The 'Inspections' section is currently selected. The main area displays a list of inspections with the following items:

- Electric Inspections
- Framing Inspection
- Mechanical Inspection
- Plumbing Inspections

- Set the **Status Reason** to **New Request** or **Pending**, change the **Scheduled Date** to today's date, and make a note of the current **Owner** of the record.

The screenshot shows the 'Electric Inspections' record details page. The 'General' tab is selected. The record has the following fields:

|                 |                             |
|-----------------|-----------------------------|
| Name            | * Electric Inspections      |
| Inspection Type | Initial Inspection          |
| Permit          | <a href="#">Test Permit</a> |
| Scheduled Date  | * 12/6/2019                 |
| Owner           | * MOD Administrator         |
| Comments        | ---                         |

- Click **Save**.
- Run the function
- Go to your **Azure** portal.
- Click **Test/Run**.

The screenshot shows the Azure Functions 'Code + Test' interface. The 'Developer' tab is selected. The 'Code + Test' section shows the following code:

```

1 {
2 "generatedBy": "Microsoft.NET.Sdk.Functions-3.0.3",

```

- Click **Run**.
- The function should run and succeed.



6. Confirm record assignment
  - Go back to the **Permit Management** application.
  - Click **Refresh**.

INSPECTION  
Electric Inspections

- The record **Owner** should now be the **Inspection Router**.

**Electric Inspections**

New Request Status Reason

**General** Related

|                 |                                     |
|-----------------|-------------------------------------|
| Name            | * Electric Inspections              |
| Inspection Type | Initial Inspection                  |
| Permit          | Test Permit                         |
| Scheduled Date  | * 12/6/2019                         |
| Owner           | * <a href="#">Inspection Router</a> |
| Comments        | ---                                 |

## 38 Exercise #4: Promote to production

**Objective:** In this exercise, you will export the Permit Management solution from your Dev environment and import it into your Production environment. In this lab, you have added a security role to the solution that must be promoted.

### 38.1 Task #1: Export Solution

1. Export Permit Management managed solution
  - Sign in to [Power Apps maker portal](#) and make sure you are in the **Dev** environment.
  - Select **Solution**.
  - Select the **Permit Management** solution and click **Export**.

The screenshot shows the 'Solutions' section of the Power Apps portal. The 'Export' button in the top navigation bar is highlighted with a red arrow. Below it, a table lists three solutions: 'PowerAppsTools\_contoso', 'Permit Management', and 'Common Data Services Default Solution'. The 'Permit Management' solution is selected.

| Display name                          | Created ↓ | Version | Managed ex |
|---------------------------------------|-----------|---------|------------|
| PowerAppsTools_contoso                | 8/30/2019 | 1.0     | ⋮          |
| Permit Management                     | 8/22/2019 | 1.0.0.2 | ⋮          |
| Common Data Services Default Solution | 8/17/2019 | 1.0.0.0 | ⋮          |

- Click **Publish** and wait for the publishing to complete.

## Before you export X

### Publish all changes

If you made changes to this solution that you'd like to export, publish them now.

[Learn more](#)

**Publish**

### Check for issues

0 found on 08/26/2019

**Next**

**Cancel**

- Click **Next**.
- Select **Managed** and click **Export**.

[← Export this solution](#) [X](#)

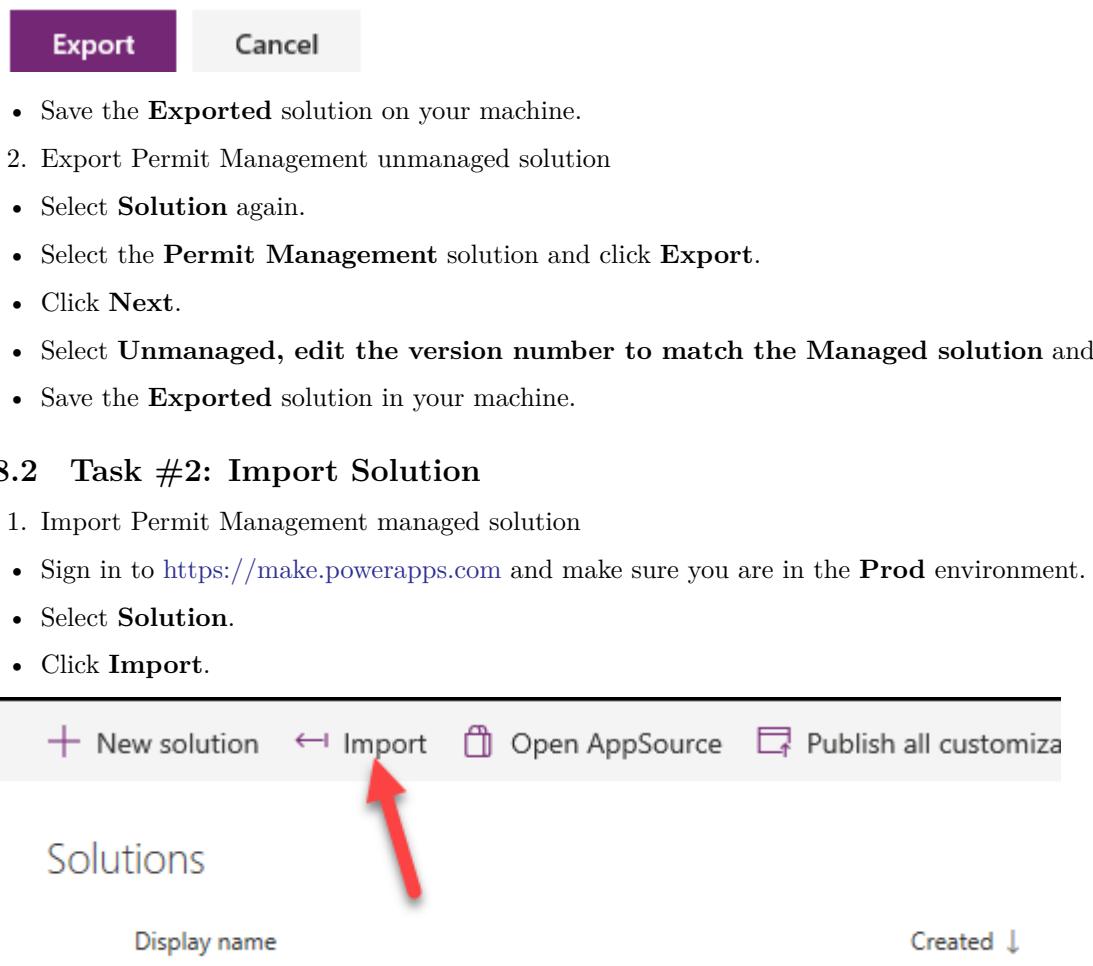
Version number\* ⓘ

Current version 1.0.0.2

Export as

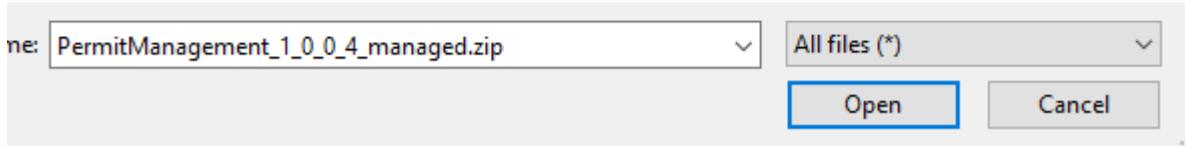
Managed (recommended) ⓘ  
The solution is moving to a test or production environment. [Learn more](#)

Unmanaged  
The solution is moving to another development environment or source control. [Learn more](#)



- Click **Choose file**.
  - Select the **Managed** solution you exported and click **Open**.

| Name                                 | Date modified      | Type                 | Size   |
|--------------------------------------|--------------------|----------------------|--------|
| PermitManagement_1_0_0_4_managed.zip | 9/10/2019 10:32 AM | Compressed (zipp...) | 108 KB |
| PermitManagement_1_0_0_5.zip         | 9/10/2019 10:34 AM | Compressed (zipp...) | 121 KB |



- Click **Next**.
- Click **Next** again.
- Click **Import**.
- Wait for the import to complete and click **Close**.
- Review and test your production environment.

### 38.3 lab: title: 'Lab 08: Publishing Events Externally'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *entity* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

### 38.4 Lab 08 – Publishing Events Externally

## 39 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will use the event publishing capability of the Microsoft Dataverse. When a permit results in changing the size of the build site, an external taxing authority needs to be notified so they can evaluate if additional taxing is required. You will configure Microsoft Dataverse to publish permits with size changes using the web hook option. To simulate the taxing authority receiving the information you will create a simple Azure function to receive the post.

## 40 High-level lab steps

As part of configuring the event publishing, you will complete the following:

- Create an Azure Function to receive the web hook post
- Configure Microsoft Dataverse to publish events using a web hook
- Test publishing of events

### 40.1 Things to consider before you begin

- Do we know what events will trigger our web hook?
- Could what we are doing with the web hook, be done using Power Automate?

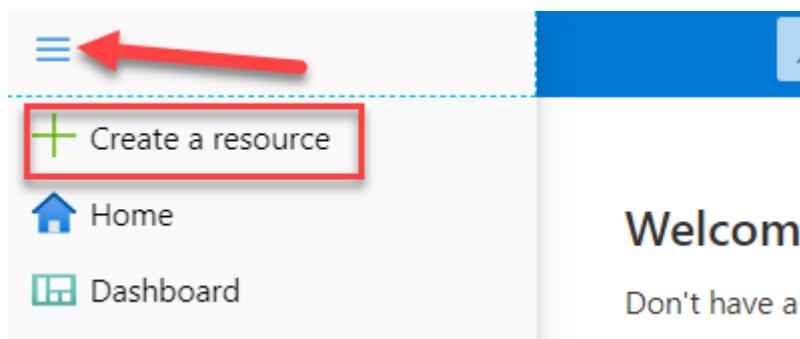
- Remember to continue working in your DEVELOPMENT environment. We'll move everything to production soon.

## 41 Exercise #1: Create an Azure Function

**Objective:** In this exercise, you will create an Azure Function that will be the endpoint to accept and log incoming web requests.

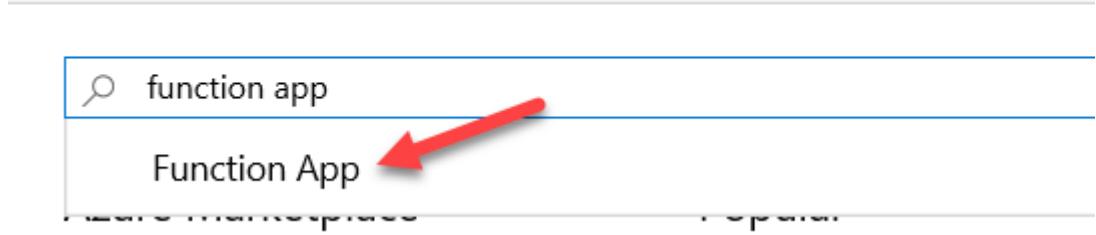
### 41.1 Task #1: Create Azure Function App

1. Create new function application
  - Sign in to [Azure portal](#) and login.
  - Click **Show portal menu** and select **+ Create a Resource**.



- Search for Function App and select it.

#### New



- Click **Create**.

#### Function App

Microsoft



#### Function App

[Save for later](#)

Microsoft

[Create](#)

- Enter your initials plus today's date for **App Name**, select your **Subscription**, select **Create New** for **Resource Group**, select **.NET Core** for Runtime Stack, select location in the same region as **Microsoft Dataverse**, and click **Review + Create**.

### Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

[REDACTED]

Resource Group \* ⓘ

MB400Resource

[Create new](#)

### Instance Details

Function App name \*

HR081420

.azurewebsites.net

Publish \*

Code  Docker Container

Runtime stack \*

.NET Core

Version \*

3.1

Region \*

Central US

[Review + create](#)

< Previous

Next : Hosting >

- Click **Create** and wait for the deployment to complete.

## 41.2 Task #2: Create an Azure Function

- Create a new function

- Click **Go to resource**.



## Your deployment is complete



Deployment name: Microsoft.Web-FunctionApp-Portal-e36b265d-...

⋮

Subscription: [REDACTED]

⋮

Resource group: MB400Resource

▼ Deployment details ([Download](#))

^ Next steps

[Go to resource](#)

- Select **Functions** and click **+ Add**.

The screenshot shows the Azure Functions portal interface. At the top, there's a search bar labeled 'Search (Ctrl+ /)' and several navigation links: '+ Add', '</> Develop Locally', 'Refresh', and 'Delete'. Below the header, a sidebar on the left lists 'Functions', 'App keys', and 'App files'. The main area is titled 'Functions' and contains a search bar 'Filter by name...' and a sorting option 'Name ↑↓'. A message 'No results' is displayed. On the left side of the main area, there's a bulleted list: '• Select **HTTP trigger**'.

## New Function

Create a new function in this function app. Start by selecting a template below.

[Templates](#)    [Details](#)

The screenshot shows the 'New Function' template selection screen. It features a search bar 'Search by template name' and two template cards. The first card, 'HTTP trigger', is highlighted with a red border. It includes an icon of a computer monitor with 'www' on it, the text 'HTTP trigger', and a description: 'A function that will be run whenever it receives an HTTP request, responding based on data in the body or query string'. The second card, 'Timer trigger', includes an icon of a clock, the text 'Timer trigger', and a description: 'A function that will be run on a specified schedule'.

- Click **Create Function** and wait for the function to be created.
2. Test the function
- Select **Code + Test**.

[fx] **HttpTrigger1** ✘

Function

Search (Ctrl+ /) « ✓ Enable ⚡ Disable 🗑 Delete 📄

**Overview**

Developer

**Code + Test** (highlighted with a red box)

Integration

- Click **Test/Run**.

Save Discard Refresh **Test/Run** (highlighted with a red box) Get function URL

HR081420 \ HttpTrigger1 \ run.csx

- Click **Run**.
- You should see **Hello, Azure** in the output.

**Input**    **Output** (highlighted with a blue underline)

#### HTTP response code

200 OK

#### HTTP response content

Hello, Azure. This HTTP triggered function executed successfully.

- Close the test pane.

### 3. Edit the function

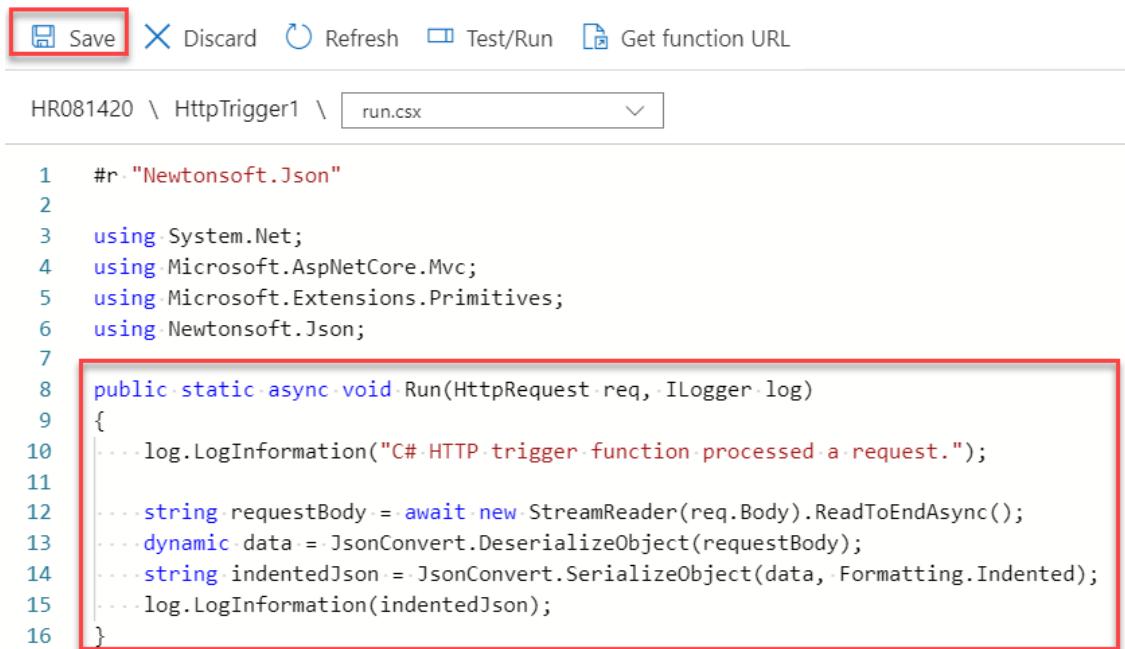
- Replace the Run method with the method below.

```
public static async void Run(HttpRequest req, ILogger log)
{
 log.LogInformation("C# HTTP trigger function processed a request.");

 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
 dynamic data = JsonConvert.DeserializeObject(requestBody);
 string indentedJson = JsonConvert.SerializeObject(data, Formatting.Indented);
 log.LogInformation(indentedJson);
```

}

- Save your changes.



```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7
8 public static async void Run(HttpRequest req, ILogger log)
9 {
10 log.LogInformation("C# HTTP trigger function processed a request.");
11
12 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
13 dynamic data = JsonConvert.DeserializeObject(requestBody);
14 string indentedJson = JsonConvert.SerializeObject(data, Formatting.Indented);
15 log.LogInformation(indentedJson);
16 }
```

#### 4. Remove HTTP output

- Select **Integration**.

#### Developer

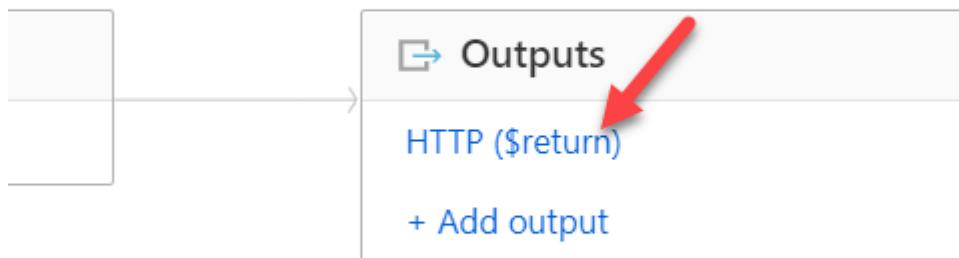
 Code + Test

 Integration

 Monitor

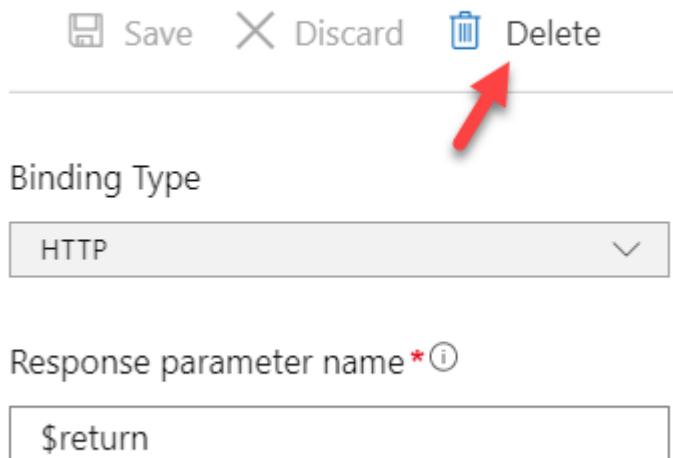
 Function Keys

- Select the **HTTP Output**.



- Click **Delete**.

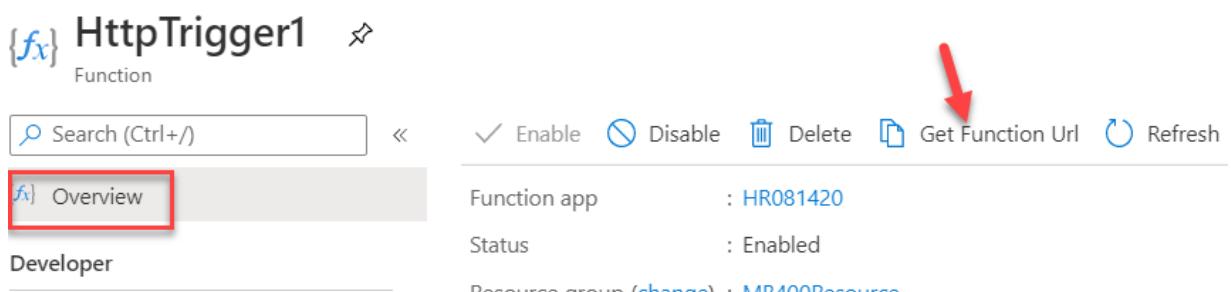
## Edit Output



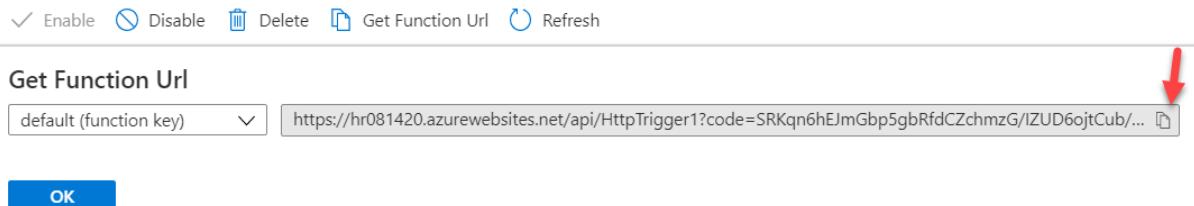
- Click **OK**.

5. Get the function URL

- Select **Overview** and click **Get Function URL**.



- Click **Copy** and click **OK** to close the popup.



- Save the **URL**, you will need it in the next exercise.

## 42 Exercise #2: Configure Web Hook

### 42.1 Task #1: Configure publishing to a web hook

1. Download the SDK Toolkit. If you already have the Plugin Registration tool from the previous lab you can proceed to step three of this task.
  - Navigate to <https://xrm.tools/SDK>
  - Click **Download SDK Zip File**.

Download SDK Tools Zip

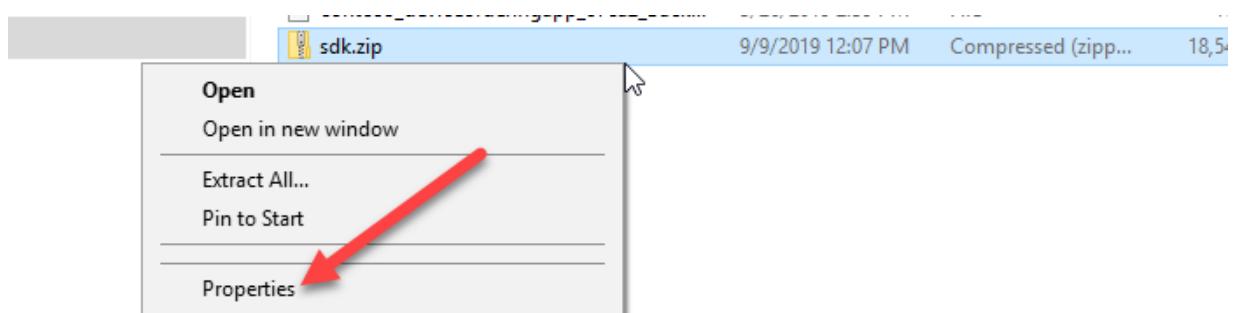
### Download a single Zip File

This zip file contains Core Tools, Plugin Registration Tool, and Package Deployer runtime

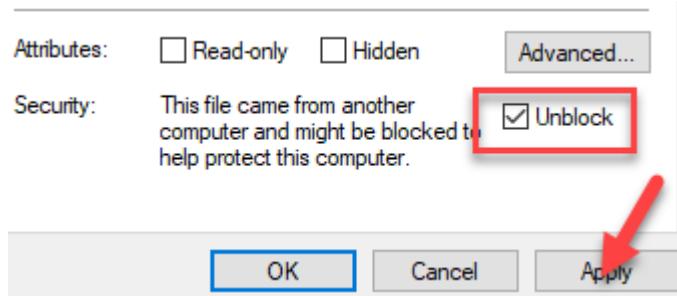
**Download SDK Zip File**

After you download, you must right-click properties and unblock the file before you unzip it locally

- Save the zip file on your machine.
- Right click on the downloaded **sdk.zip** file and select **Properties**.



- Check the **Unblock** checkbox and click **Apply**.



- Click **OK**.
- Right click on the  **sdk.zip** file again and select **Extract All**.
- Complete extracting.

## 2. Start the Plugin Registration Tool

- Open the  **sdk** folder you extracted and click to open the **PluginRegistration** folder.

|                                    |                   |                    |
|------------------------------------|-------------------|--------------------|
| <a href="#">ConfigMigration</a>    | 9/9/2019 12:11 PM | File folder        |
| <a href="#">CoreTools</a>          | 9/9/2019 12:11 PM | File folder        |
| <a href="#">PackageDeployment</a>  | 9/9/2019 12:11 PM | File folder        |
| <a href="#">PluginRegistration</a> | 9/9/2019 12:11 PM | File folder        |
| <a href="#">readme.txt</a>         | 9/9/2019 6:02 AM  | Text Document 1 KB |

- Locate and double click **PluginRegistration.exe**.
- 
- |                                                 |         |
|-------------------------------------------------|---------|
| <a href="#">PluginRegistration.exe</a>          | 7/16/20 |
| <a href="#">PluginProfiler.Plugins.dll</a>      | 7/16/20 |
| <a href="#">PluginProfiler.Solution.zip</a>     | 7/16/20 |
| <a href="#">PluginRegistration.exe</a>          | 7/16/20 |
| <a href="#">PluginRegistration.exe.config</a>   | 7/16/20 |
| <a href="#">ServiceEndpointRegistration.dll</a> | 7/16/20 |

## 3. Create new connection

- Click **Create New Connection**.



- Select **Office 365** and check the **Display List of available organization** and **Show Advanced** checkboxes. Select **Online Region** where your organization is located. If you are unsure what region to select, select **Don't Know**.
- Provide your **Microsoft Dataverse** credentials and click **Login**.

## Login

Deployment Type:  On-premises  Office 365

Sign in as current user  
 Display list of available organizations  
 Show Advanced

Advanced

|               |                                   |
|---------------|-----------------------------------|
| Online Region | Don't Know                        |
| User Name     | admin@M365x920496.onmicrosoft.com |
| Password      | *****                             |

**Login** **Cancel**

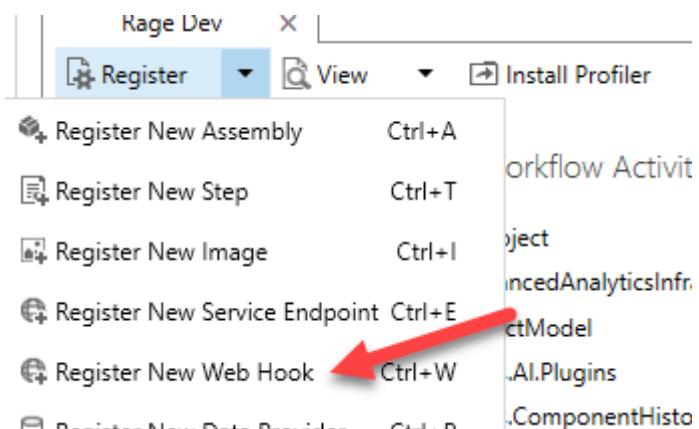
- Select the **Dev** environment and click **Login**.

| CRM Region    | Organization Name       |
|---------------|-------------------------|
| North America | Rage Dev - org8a78478f  |
| North America | Rage Prod - org00da204e |

**Login** **Cancel**

#### 4. Register new Web Hook

- Click **Register** and select **Register New Web Hook**.



- Enter **NewSize** for **Name**.
- Go to the notepad where you saved the function URL and copy everything before the '?'.

[https://hr09092019.azurewebsites.net/api/HttpTrigger1?  
code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==](https://hr09092019.azurewebsites.net/api/HttpTrigger1?code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==)

- Go back to the **Plugin Registration** tool and paste the **URL** you copied in the **Endpoint URL** Column.

## WebHook Registration

|                |                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name           | NewSize                                                                                                                                                                                                                                                    |
| Endpoint URL   | <a href="https://hr09092019.azurewebsites.net/api/HttpTrigger1?code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==">https://hr09092019.azurewebsites.net/api/HttpTrigger1?<br/>code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==</a> |
| Authentication | HttpHeader                                                                                                                                                                                                                                                 |

- Select **WebhookKey** for **Authentication**.
- Go back to the notepad and copy the key.

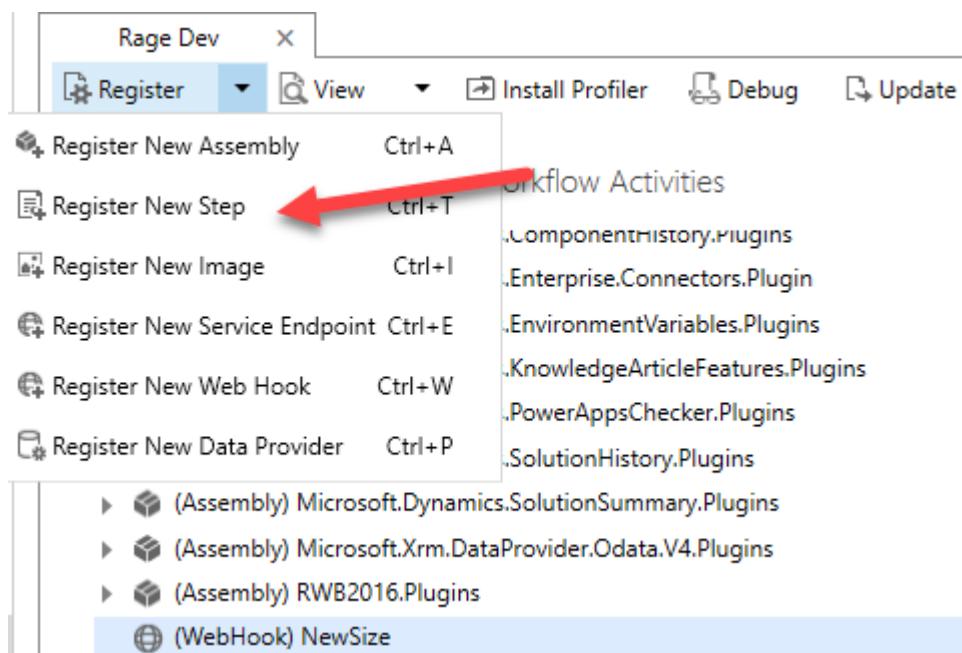
[https://hr09092019.azurewebsites.net/api/HttpTrigger1?  
code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==](https://hr09092019.azurewebsites.net/api/HttpTrigger1?code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==)

- Go back to the **Plugin Registration** tool, paste the key you copied in the **Value** Column and click **Save**.

## WebHook Registration

|                |                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name           | NewSize                                                                                                                                                                                                                                                    |
| Endpoint URL   | <a href="https://hr09092019.azurewebsites.net/api/HttpTrigger1?code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==">https://hr09092019.azurewebsites.net/api/HttpTrigger1?<br/>code=ecVN3/JiBMDljpAn8W/xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==</a> |
| Authentication | WebhookKey                                                                                                                                                                                                                                                 |
| Value          | /xk1/oNose4NUFs14oexaBI9vasKcw3uksvg==                                                                                                                                                                                                                     |

5. Register new step
  - Select the **Web Hook** you registered, click **Register** and select **Register New Step**.



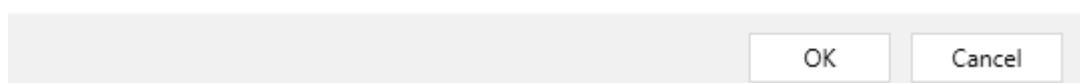
- Select **Update** for Message, **contoso\_permit** for Primary Table, and click **Filtering Attributes**.

#### General Configuration Information

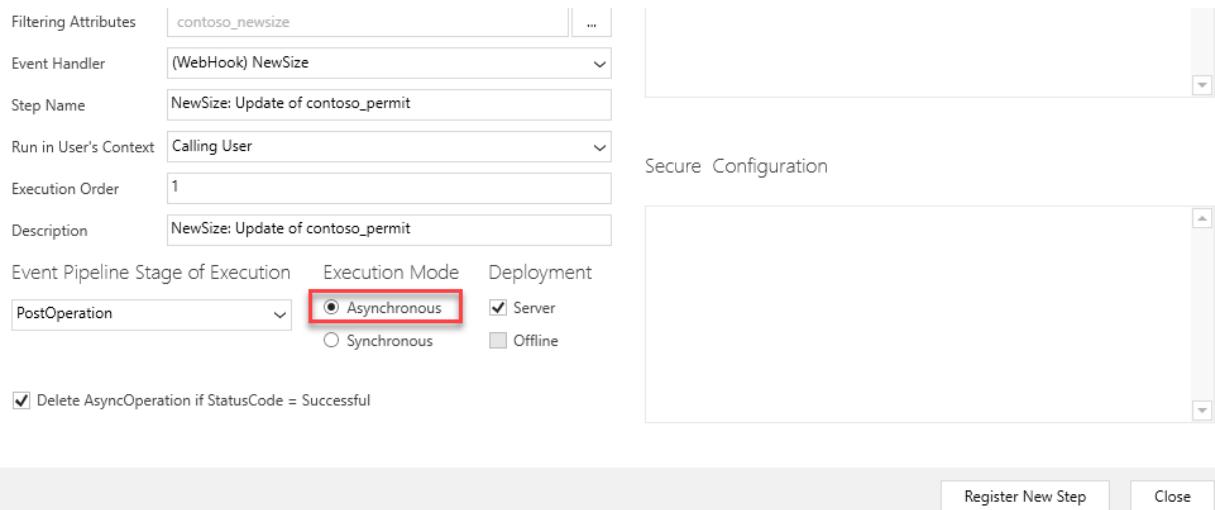
|                      |                                                                                  |
|----------------------|----------------------------------------------------------------------------------|
| Message              | <input type="text" value="Update"/>                                              |
| Primary Entity       | <input type="text" value="contoso_permit"/>                                      |
| Secondary Entity     | <input type="text"/>                                                             |
| Filtering Attributes | <input type="text" value="All Attributes"/> <span style="color: red;">...</span> |
| Event Handler        | <input type="text" value="(WebHook) NewSize"/>                                   |

- Select only **New Size** and click **OK**.

|                                                 |                    |          |
|-------------------------------------------------|--------------------|----------|
| <input type="checkbox"/> Modified By            | modifiedby         | Lookup   |
| <input type="checkbox"/> Modified By (Delegate) | modifiedonbehalfby | Lookup   |
| <input type="checkbox"/> Modified On            | modifiedon         | DateTime |
| <input type="checkbox"/> Name                   | contoso_name       | String   |
| <input checked="" type="checkbox"/> New Size    | contoso_newsize    | Integer  |
| <input type="checkbox"/> Owner                  | ownerid            | Owner    |
| <input type="checkbox"/> Owning Business Unit   | owningbusinessunit | Lookup   |



- Select **Asynchronous** for Execution Mode and click **Register New Step**.



## 42.2 Task #2: Test the Web Hook

### 1. Start the Permit Management application

- Sign in to [Power Apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select Apps and click to open the Permit Management application.

- Select **Permits** and open one of the permit records. Create new if you don't have a Permit record.

- Change the **New Size** to **5000** and **Save**.

|             |                                                         |
|-------------|---------------------------------------------------------|
| Name        | * Test Permit                                           |
| Permit Type | <input checked="" type="checkbox"/> New Construction    |
| Build Site  | <input checked="" type="checkbox"/> One Microsoft Way   |
| Contact     | <input checked="" type="checkbox"/> John Doe            |
| Start Date  | * 8/30/2019 <input type="button" value="Calendar"/>     |
| New Size    | * 5000                                                  |
| Owner       | * <input checked="" type="checkbox"/> MOD Administrator |



## 2. Check Azure Output

- Go back to your **Azure Function**.
- Select **Code + Test**.
- Show **Logs**.



- You should see logs like the image below. The Output is a serialized **RemoteExecutionContext** object

```
2020-08-14T16:18:30Z [Information] Executing 'Functions.HttpTrigger1' (Reason='This
19c9-8c6c-51d84facaa90)
2020-08-14T16:18:30Z [Information] C# HTTP trigger function processed a request.
2020-08-14T16:18:30Z [Information] {
 "BusinessUnitId": "0c543eea-65d9-ea11-a819-000d3a30f81d",
 "CorrelationId": "cb91b17d-b5a4-48a5-9b64-4b5aa7c6c9b2",
 "Depth": 1,
 "InitiatingUserAzureActiveDirectoryObjectId": "00000000-0000-0000-0000-000000000000",
 "InitiatingUserId": "0c1dd508-085a-4e63-8292-e42cb5e1563e",
 "InputParameters": [
 {
 "key": "Target",
 "value": {
 "__type": "Entity:http://schemas.microsoft.com/xrm/2011/Contracts",
 "Attributes": [
 {
 "key": "contoso_newsize",
 "value": 5000
 },
 {
 "key": "contoso_permitid",
 "value": "6536bd95-17dc-ea11-a816-00224803d22e"
 }
]
 }
 }
]
}
```

**Hint:** If the log is not showing in the console (sometimes this happens), click **Monitor** on the left and check execution log. Select entry, details will be on the right (this could be delayed up to a few minutes).

6. Confirm the function executes only when the New Size value changes

- Go back to the **Permit Management** application.
- Change the **Start Date** to tomorrow's date and click **Save**.

The screenshot shows the 'General' tab of a 'Test Permit' record in the 'PERMIT' entity. The 'Name' field is set to 'Test Permit'. The 'Permit Type' is 'New Construction' and the 'Build Site' is 'One Microsoft Way'. The 'Contact' is 'John Doe'. The 'Start Date' is set to '9/10/2019'. The 'New Size' is '5,000'. The 'Owner' is 'MOD Administrator'. The status is 'Locked'. At the bottom, there is a 'Save' button with a red arrow pointing to it.

|             |                   |
|-------------|-------------------|
| Name        | * Test Permit     |
| Permit Type | New Construction  |
| Build Site  | One Microsoft Way |
| Contact     | John Doe          |
| Start Date  | * 9/10/2019       |
| New Size    | * 5,000           |
| Owner       | MOD Administrator |

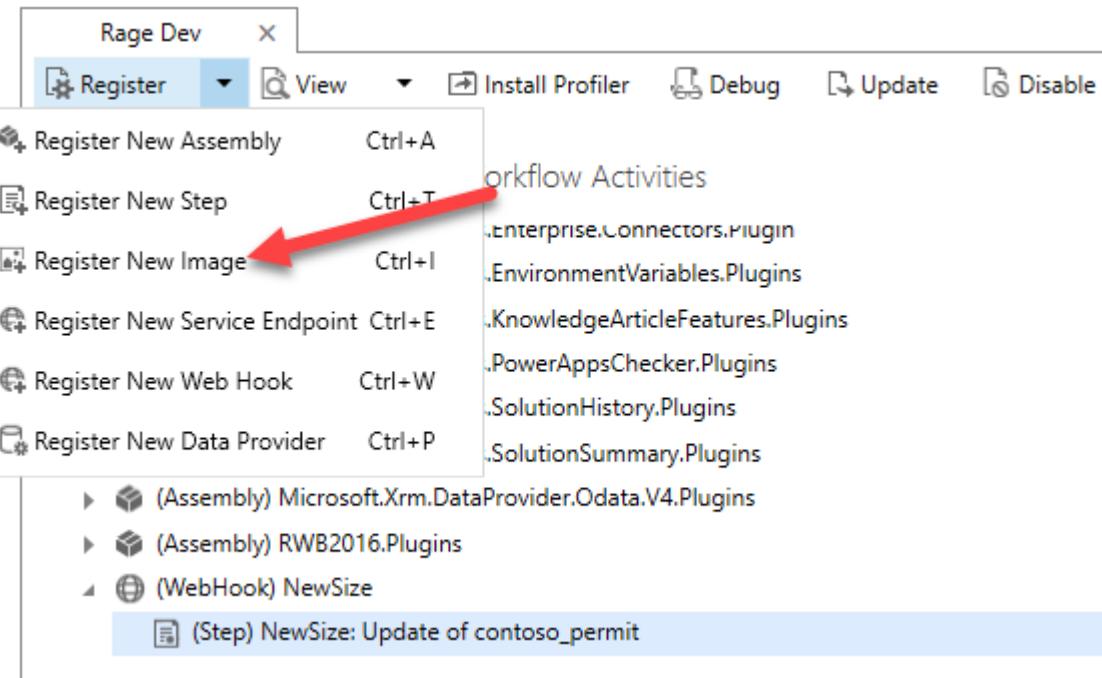
Go back to the Azure Function and make sure the function did not execute.

#### 42.3 Task #3: Configure an Table image

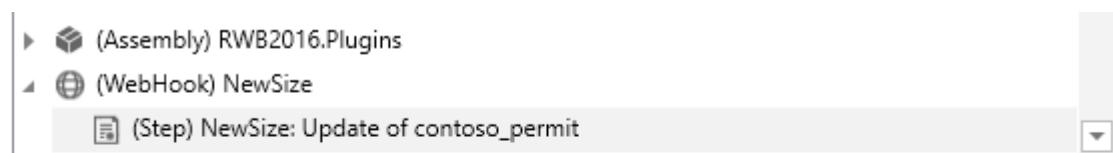
This step allows you to avoid unnecessarily querying Microsoft Dataverse and make a request only when you need information from the primary Table. It can also be used to get the prior value of a Column before an update operation.

##### 1. Register New Image

- Go back to the **Plugin Registration** tool.
- Select the **NewSize** step you created, click **Register** and select **Register New Image**.



- Check both **Pre** and **Post** images checkboxes.
- Enter **Permit Image** for **Name**, **PermitImage** for **Table Alias**, and click on the **Parameters** button.



#### Image Type

|                                               |                                                       |
|-----------------------------------------------|-------------------------------------------------------|
| <input checked="" type="checkbox"/> Pre Image | <input checked="" type="checkbox"/> Post Image        |
| Name                                          | Permit Image                                          |
| Entity Alias                                  | PermitImage                                           |
| Parameters                                    | All Attributes <span style="float: right;">...</span> |

Register Image Cancel

- Select **Build Site**, **Contact**, **Name**, **New Size**, **Permit Type**, and **Start Date**, and then click **OK**.

|                                                                 |                     |                  |
|-----------------------------------------------------------------|---------------------|------------------|
| <input checked="" type="checkbox"/> <a href="#">Name</a>        | contoso_name        | String           |
| <input checked="" type="checkbox"/> <a href="#">New Size</a>    | contoso_newsize     | Integer          |
| <input type="checkbox"/> <a href="#">Owner</a>                  | ownerid             | Owner            |
| <input type="checkbox"/> <a href="#">Owning Business Unit</a>   | owningbusinessunit  | Lookup           |
| <input type="checkbox"/> <a href="#">Owning Team</a>            | owningteam          | Lookup           |
| <input type="checkbox"/> <a href="#">Owning User</a>            | owninguser          | Lookup           |
| <input type="checkbox"/> <a href="#">Permit</a>                 | contoso_permitid    | Uniqueidentifier |
| <input checked="" type="checkbox"/> <a href="#">Permit Type</a> | contoso_permittype  | Lookup           |
| <input type="checkbox"/> <a href="#">Record Created On</a>      | overriddencreatedon | DateTime         |
| <input checked="" type="checkbox"/> <a href="#">Start Date</a>  | contoso_startdate   | DateTime         |
| <input type="checkbox"/> <a href="#">Status</a>                 | statecode           | State            |

- Click Register Image.

#### Image Type

Pre Image       Post Image

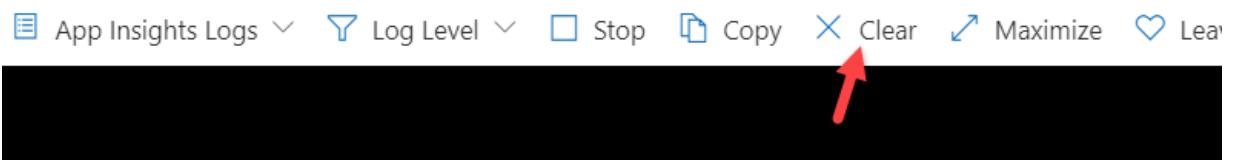
Name

Entity Alias

Parameters  ...

#### 2. Clear Azure log

- Go back to the **Azure Function**.
- Clear Logs.

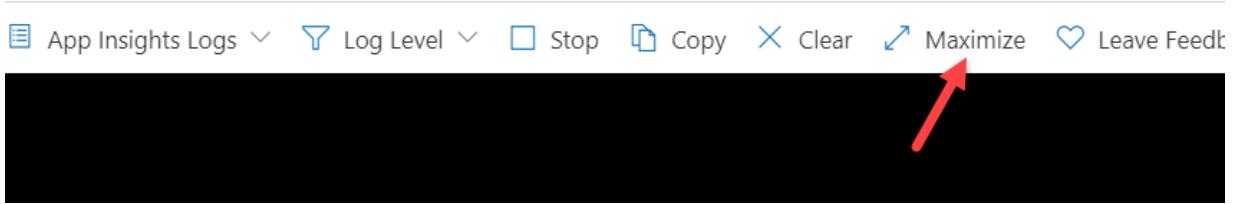


#### 3. Update Permit record

- Go to the **Permit Management** application.
- Select **Permits** and open one of the **Permit** records.
- Change the New Size to **4000** and click **Save**

#### 4. Check Azure logs

- Go back to the **Azure Function**.
- Maximize the log pane.



- The logs should now show both **Pre** and **Post** Table images. In this case you should see the old value **5000** in **Pre** image and the new value **4000** in the **Post** image

```
"FormattedValues": [
 {
 "key": "contoso_buildsite",
 "value": "One Microsoft Way"
 },
 {
 "key": "contoso_contact",
 "value": "John Doe"
 },
 {
 "key": "contoso_newsize",
 "value": "5,000"
 },
 {
 "key": "contoso_permittype",
 "value": "New Construction"
 },
 ...
],
"EntityState": null,
"FormattedValues": [
 {
 "key": "contoso_buildsite",
 "value": "One Microsoft Way"
 },
 {
 "key": "contoso_contact",
 "value": "John Doe"
 },
 {
 "key": "contoso_newsize",
 "value": "4,000"
 },
 {
 "key": "contoso_permittype",
 "value": "New Construction"
 },
 ...
]
```

**42.4 Note:** Technically, we have the data in the target object already. However, if there are plugins modifying the data, PostImage will contain the copy as recorded in Microsoft Dataverse while Target contains the data as submitted on Save. In addition to that, preimage contains data before the save operation took place.

#### 42.5 lab: title: 'Lab 09: Custom Connector'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *Table* is now *table* and *Column* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

#### 42.6 Lab 09 – Custom Connector

### 43 Scenario

A regional building department issues and tracks permits for new buildings and updates for remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will build a custom connector that can be used from Power Apps and Power Automate. Custom connectors describe existing APIs and allow them to be used easily. In this lab, you will build an API that has common calculations used by inspectors so that they can be used by applications. After building the API, you will create a custom connector definition to make it available to Power Apps and Power Automate.

The connector could have multiple actions defined on it. However, for our lab we will define a single action **Get Required CPM** – where CPM stands for Cubic Per Minute. In some regions this would be Cubic Feet Per Minute, and in others it could be Cubic Meters Per Minute. The action we are building will take the dimensions of a room and the number of air exchanges required by policy. The action logic will calculate the required CPM for that configuration. If you want, you can add additional actions to support other inspection type scenarios to the API and the custom connector.

After building the API and the custom connector you will modify the inspector app to add the user experience to use the connector. You will use the same connector and invoke an action from Power Automate.

## 44 High-level lab steps

As part of configuring the custom connector, you will complete the following

- Create an Azure function that implements the CPM API
- Create a custom connector in a solution
- Configure the security and actions on the custom connector
- Test the custom connector
- Configure a Power Apps canvas app to use the connector
- Configure a Power Automate to use the connector

### 44.1 Things to consider before you begin

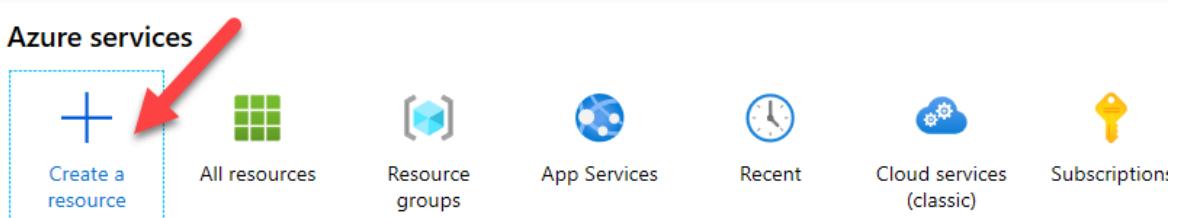
- Is there a standard approved connector already available that can be used?
- What security will we use in our connector?
- What are possible triggers and actions of the connector?
- Are there any API rate limits that could potentially affect the connector?

## 45 Exercise #1: Create the Azure Function

**Objective:** In this exercise, you will create an Azure function app and the function that will calculate the CPM.

### 45.1 Task #1: Create CPM Calculation Function

1. Create the function app
  - Sign in to [Azure portal](#)
  - Click **Create a Resource**.

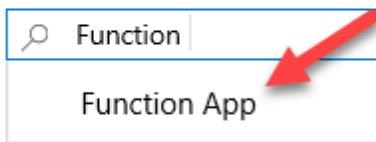


#### Recent resources

- Search for **Function** and select **Function App**.

Home > New

## New



Windows Server 2016 Datacenter

- Click Create.

## Function App

Microsoft



## Function App

Microsoft

Save for later

Create

- Enter **CPMCalculator** for **App Name**. Provide a unique name, if CPMCalculator is not available.
- Create **New Resource Group**. You may select an existing Resource Group if you already created one for this course.
- Select **.NET Core** for **Runtime Stack**, select your **Region** and click **Review + Create**.

## Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

[REDACTED]

Resource Group \* ⓘ

MB400Resource

[Create new](#)

## Instance Details

Function App name \*

CPMCalculator1

.azurewebsites.net

Publish \*

Code  Docker Container

Runtime stack \*

.NET Core

Version \*

3.1

Region \*

Central US

[Review + create](#)

< Previous

Next : Hosting >

- Click **Create**. Wait for the function app to be created

## 2. Create function

- Click **Go to Resource**.

 Delete  Cancel  Redeploy  Refresh



## Your deployment is complete



Deployment name: Microsoft.Web-FunctionApp-Portal-a94

Subscription: [REDACTED]

Resource group: hr120519

Deployment details [\(Download\)](#)

Next steps

[Go to resource](#)

- Select **Functions** and click **+ Add**.

CPMCalculator1 | Functions  
App Service

Search (Ctrl+/) <> **+ Add** Develop Locally Refresh Delete

⚡ Events (preview)

Functions

Functions

💡 App keys

Name ↑

No results.

- Select **HTTP trigger**.
- Click **Create Function**.

3. Add the **Using Statements** and **CRMCalcRequest** class to the function.

- Select **Code + Test**.

Overview

Fur

Sta

Res

Sul

Sul

**Developer**

**Code + Test**

Integration

Monitor

- Add the Using Statements below to the function.

```
using Microsoft.Extensions.Logging;
using Newtonsoft.Json.Linq;
```

run.csx

Save

▶ Save and run

</> Get function URL

```
1 #r "Newtonsoft.Json"
2
3 using System.Net;
4 using Microsoft.AspNetCore.Mvc;
5 using Microsoft.Extensions.Primitives;
6 using Newtonsoft.Json;
7 using Microsoft.Extensions.Logging;
8 using Newtonsoft.Json.Linq;
9
10 public static async Task<IActionResult> Run(HttpContext req, ILogger lo
```

- Add the public class below to the function. This will describe the request that will be sent from the applications using the API.

```
public class CRMCalcRequest
{
 public int Width=0;
 public int Height=0;
 public int Length=0;
```

```

 public int AirChanges=0;
 }

17 return name != null
18 ? (ActionResult)new OkObjectResult($"Hello, {name}")
19 : new BadRequestObjectResult("Please pass a name on the query string or in the request body")
20 }
21
22
23 public class CPMCalcRequest
24 {
25
26 public int Width=0;
27 public int Height=0;
28 public int Length=0;
29 public int AirChanges=0;
30
31 }
32
33

```

4. Clean up the Run method

- Go to the **Run** method.
- Remove everything but the log line from the **Run** method.

```

public static async Task<IActionResult> Run(HttpRequest req, ILogger log)
{
 log.LogInformation("C# HTTP trigger function processed a request.");
}

```

5. Get the Request body and deserialize it as **CPMCalcRequest**

- Get the request **Body** from the request argument. Add the code below inside the **Run** method.
 

```
string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
```
- Deserialize the request body. Add the code below inside the **Run** method.
 

```
CPMCalcRequest calcReq = JsonConvert.DeserializeObject<CPMCalcRequest>(requestBody);
```

```

8 public static async Task<IActionResult> Run(HttpRequest req, ILogger log)
9 {
10 log.LogInformation("C# HTTP trigger function processed a request.");
11
12 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
13 CPMCalcRequest calcReq = JsonConvert.DeserializeObject<CPMCalcRequest>(requestBody);
14 }
15

```

6. Calculate the CPM and return it form the Run method

- Calculate the **CPM** and log the calculated value. Add the code below inside the **Run** method.
 

```
var cpm = ((calcReq.Width*calcReq.Height*calcReq.Length) * calcReq.AirChanges) / 60;
log.LogInformation("CPM " + cpm);
```

- Return the calculated **CPM**. Add the code below inside the Run method.

```

return (ActionResult)new OkObjectResult(new{
 CPM = cpm
});
```

```

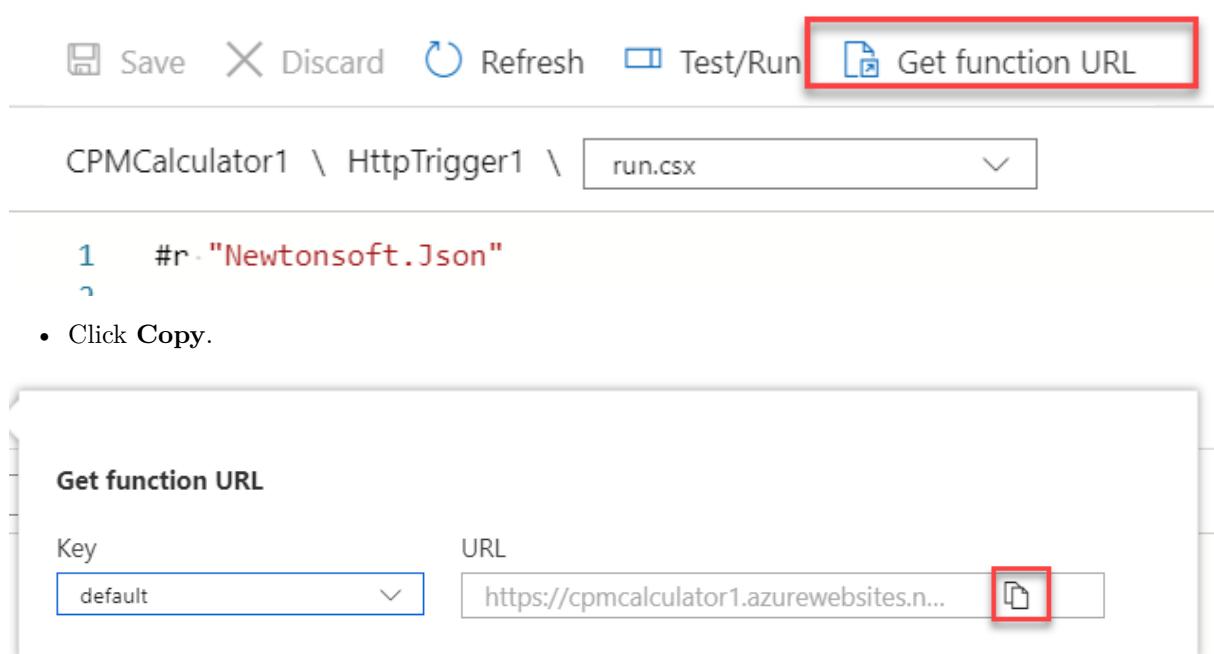
/
8 public static async Task<IActionResult> Run(HttpRequest req, ILogger log)
9 {
10 log.LogInformation("C# HTTP trigger function processed a request.");
11
12 string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
13 CPMCalcRequest calcReq = JsonConvert.DeserializeObject<CPMCalcRequest>(requestBody);
14
15 var cpm = ((calcReq.Width*calcReq.Height*calcReq.Length) * calcReq.AirChanges) / 60;
16 log.LogInformation("CPM " + cpm);
17
18 return (ActionResult)new OkObjectResult(new{
19 CPM = cpm
20 });
21 }

```

7. Click **Save** to save your changes.

8. Copy the Function URL.

- Click **Get Function URL**.



- Click **Copy**.

- Keep the URL you copied on a notepad. You will need this URL while creating the custom connector.

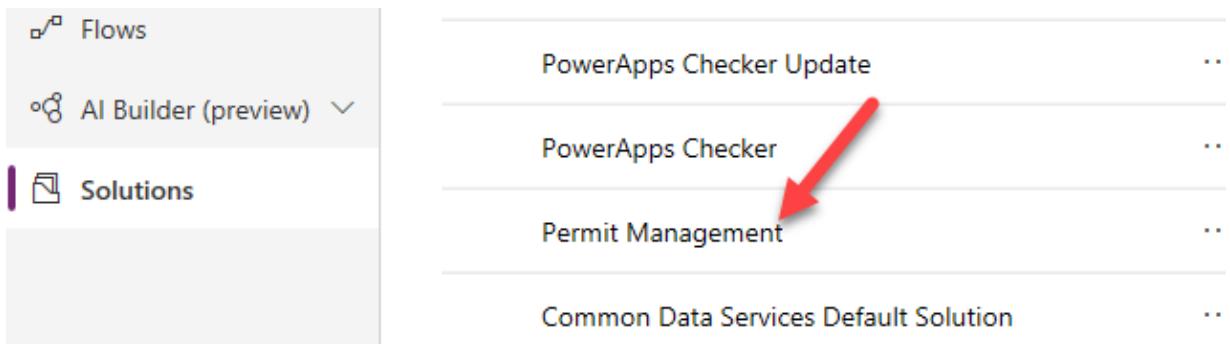
## 46 Exercise #2: Create the Custom Connector

**Objective:** In this exercise, you will create the Custom Connector. This same approach could be used to describe any existing API you create or that has been created by any 3rd parties and an existing connector for that service is unavailable.

### 46.1 Task #1: Create the Custom Connector

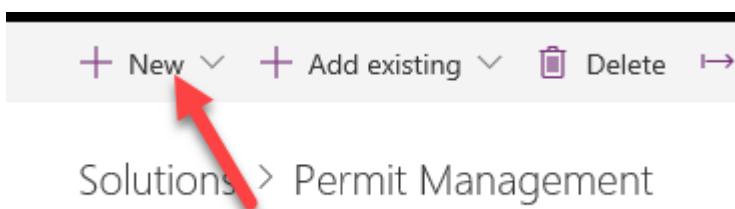
1. Open the Permit Management solution

- Sign in to [Power Apps maker portal](#) and make sure you are in the **Dev** environment.
- Select **Solutions**.
- Click to open the **Permit Management** solution.

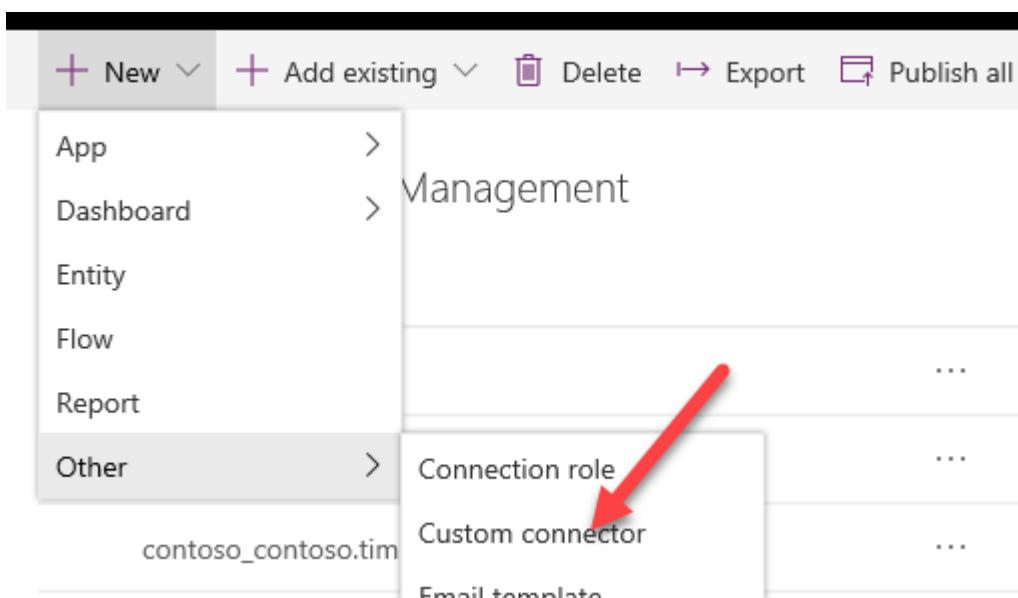


2. Create Custom Connector

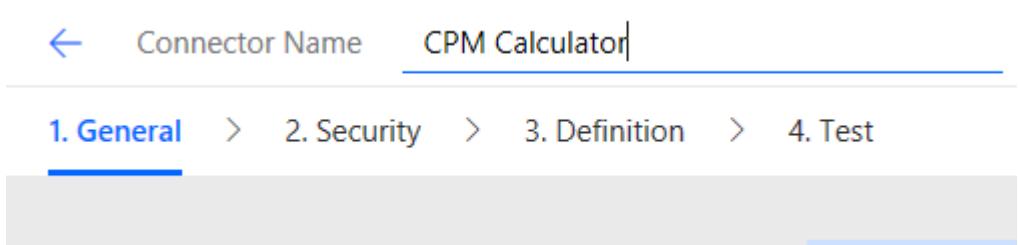
- Click **+ New**.



- Select **Other| Custom Connector**.



- Change the **Connector Name** from Untitled to **CPM Calculator**.



- Locate the **Host** Column and paste the **Function URL** you copied in Exercise 1.
- Remove https:// and everything after .net.

Scheme \*

HTTPS  HTTP

Host \*

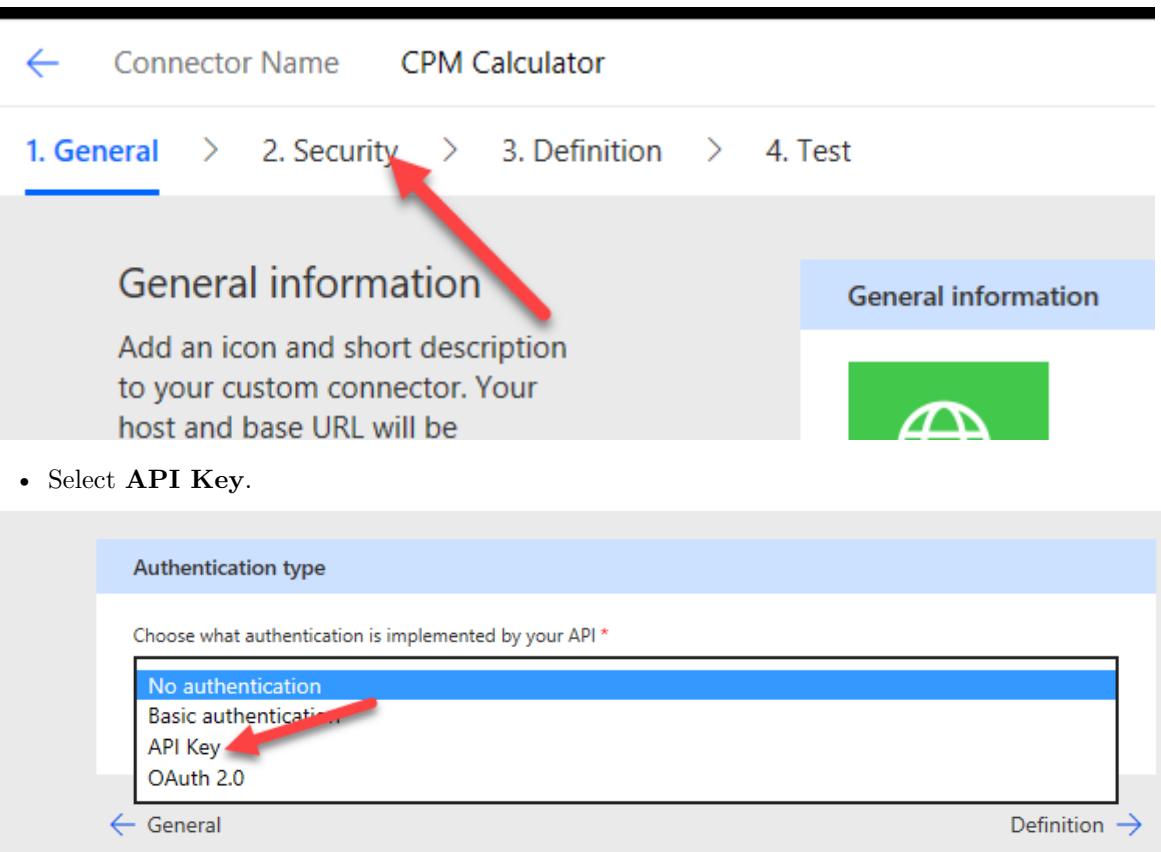
cpmcalculator.azurewebsites.net

Base URL

---

3. Select API key for security and create the connector

- Advance to **Security**.



Connector Name: CPM Calculator

1. General > 2. Security > 3. Definition > 4. Test

### General information

Add an icon and short description to your custom connector. Your host and base URL will be

General information

Authentication type

Choose what authentication is implemented by your API \*

- No authentication
- Basic authentication
- API Key** (highlighted with a red arrow)
- OAuth 2.0

← General      Definition →

- Enter **API Key** for **Parameter Label**, **code** for **Parameter Name**, and select **Query** for **Parameter Location**.

**API Key**

Users will be required to provide the API Key when creating a connection

Parameter label \*

Parameter name \*

Parameter location \*

 Edit

[← General](#) [Definition →](#)

#### 4. Create Connector

- Advance to **Definition**.

Parameter location \*

 Edit

[← General](#) [Definition →](#)

- Click **Create Connector** and wait for the connector to be created.

[3. Definition](#) > [4. Test](#) [!\[\]\(5b6d4b7d99f8dd663703f33128fa18b1\_img.jpg\) Create connector](#) [!\[\]\(061b9941b88e094712d193921e6ff316\_img.jpg\) Cancel](#)

Operations  
Actions  
ate,  
in the

 Start by adding an action or trigger on the left.

[Security](#) [Test →](#)

#### 5. Create Action

- Click **New Action**. The action describes each operation that the API has. These can be manually defined like we are doing here or can be imported from Open API or Postman collection files for larger APIs.

### ▽ Actions (0)

Actions determine the operations that users can perform. Actions can be used to read, create, update or delete resources in the underlying connector.

(i) Start by adding an action or tri

← Security

 **New action**

- Enter **CPM Calculator** for **Summary**, **Calculates CPM** for **Description**, and **GetRequiredCPM** for **Operation ID**.

#### General

Summary [Learn more](#)

CPM Calculator

Description [Learn more](#)

Calculates CPM

Operation ID \*

This is the unique string used to identify the operation.

GetRequiredCPM

Visibility [Learn more](#)

none  advanced  internal  important

#### 6. Import request from sample

- Click **+** **Import from Sample**.

#### Request

It defines the pre-requirements needed in order to make a request. Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

 **Import from sample**

- Select **Post** for **Verb**.
- Paste the function **URL** from your notepad and remove everything after **HttpTrigger1**.

## Import from sample

X

Verb \*

- GET  DELETE  POST  PUT  HEAD  
 OPTIONS  PATCH

URL \*

`https://cpmccalculator.azurewebsites.net/api/HttpTrigger1`

This is the request URL.

Headers

Headers separated by a new line, e.g.:  
Content-Type application/json  
Accept application/json

- Paste the json below in the **Body** field and click **Import**.

```
{
 "Width": 15,
 "Height": 8,
 "Length":20,
 "AirChanges":8
}
```

These are custom headers that are part of the request.

Body

```
{
 "Width": 15,
 "Height": 8,
 "Length":20,
 "AirChanges":8
}
```

The body is the payload that's appended to the HTTP request. There can only be one body parameter.

**Import**

**Close**

## 7. Add Default response

- Scroll down to the **Response** section and click + **Add Default Response**.

## Response

It defines the shape of response returned by the underlying connector.

default default

[+ Add default response](#)

- Paste the json below in the **Body** and click **Import**.

```
{"cpm":200}
```

Accept application/json

These are custom headers that are part of the response.

### Body

```
{"cpm":200}
```

The payload that is available on the response. These are the tokens that will show up as the outputs in designer.

[Import](#)

[Close](#)

- Click **Update Connector**.

[✓ Update connector](#) [X Close](#)

[+ Import from sample](#)

## 8. Test the connector

- Advance to Test.

[←](#) Connector Name

CPM Calculator 1

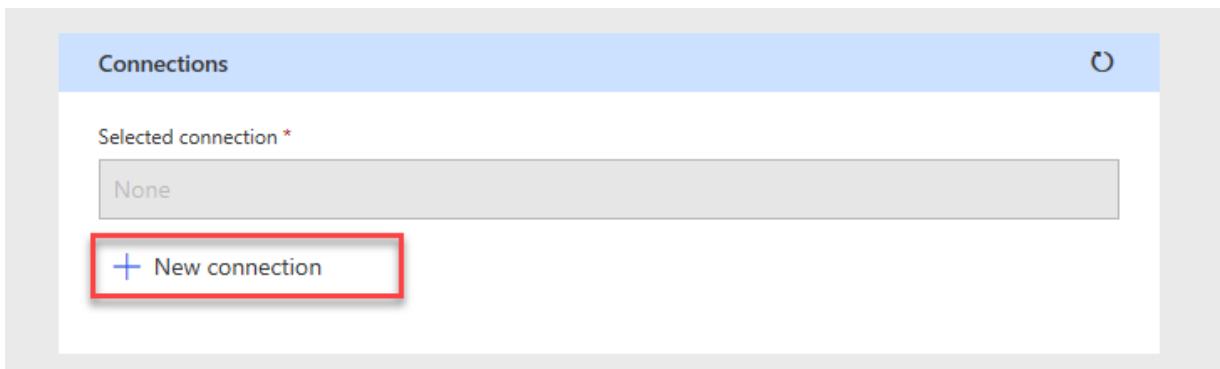
1. General > 2. Security > 3. Definition > 4. Test

[▽ Actions \(1\)](#)

Actions determine the

General

- Click **New Connection**. This will open a New window.



- Go to your notepad and copy only the value of the **code**.

`https://cpmccalculator.azurewebsites.net/api/HttpTrigger1?code=W1RANA62qDDoi6XuG/4D03HWJbQ3zCzTora3edXqr02lCEdKkpocaA==`

- Go back to the connector, paste the value you copied, and click **Create Connection**.

[← Add CPM Calculator 1 connection](#)

API Key \*

\*\*\*\*\*

Cancel

Create connection

- Refresh the connections and select newly created connection.
- Provide values for **Width**, **Height**, **Length**, and **AirChanges**.
- Click **Test Operation**.

## GetRequiredCPM

Raw Body



Width

15

Height

8

Length

15

AirChanges

5

**Test operation**

- You should get a CPM value back.

### Request

### Response

#### Status

(200)

#### Headers

```
{
 "content-length": "132",
 "content-type": "application/json; charset=utf-8",
 "date": "Wed, 18 Sep 2019 21:46:55 GMT",
 "request-context": "appId=cid-v1:4eacc8b1-a161-431e-8a75-e8fdb3012fc7"
}
```

#### Body

```
{
 "cpm": 150
}
```

- Close the window and go back to Solution window and click **Done**.
- Click **Publish all Customizations**.

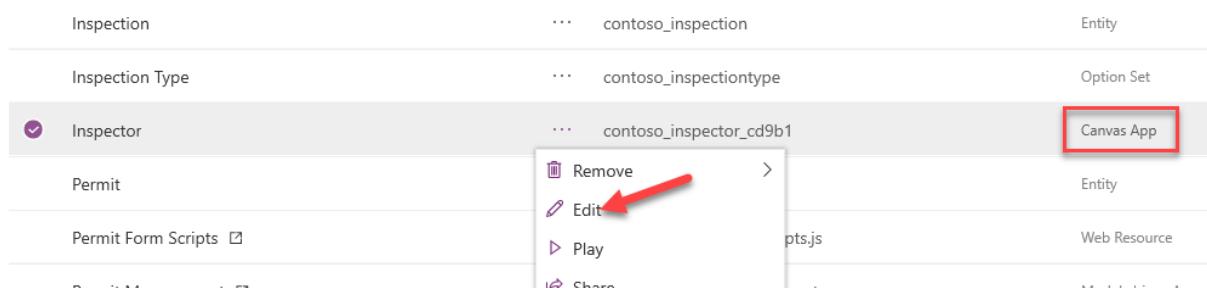
## 47 Exercise #3 Test Connector

**Objective:** In this exercise, you will use the Custom Connector from a Power 9Apps canvas app and a Power Automate.

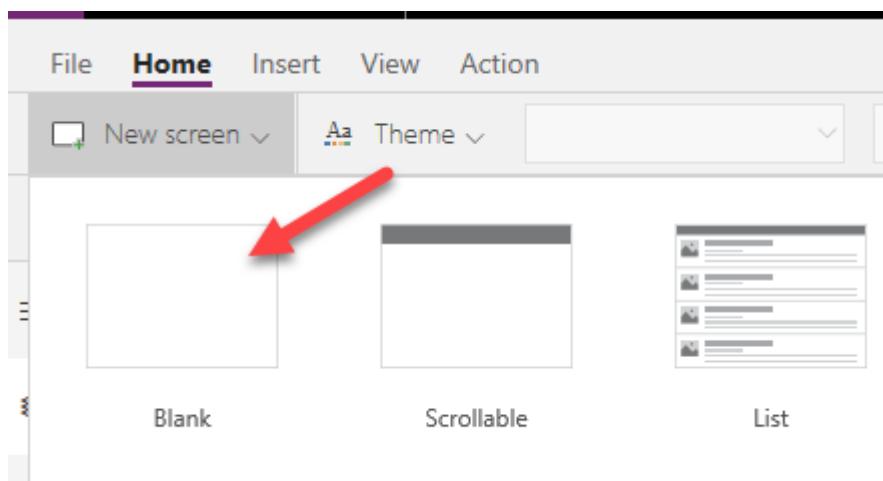
### 47.1 Task #1: Test on Canvas App

1. Open the Permit Management solution

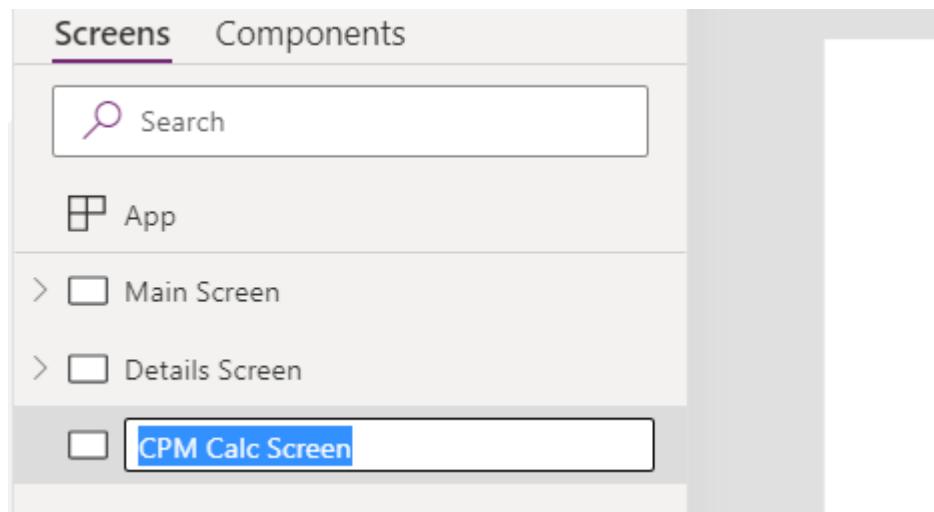
- Sign in to [Power Apps maker portal](#) and make sure you are in the **Dev** environment.
  - Select **Solutions**.
  - Click to open the **Permit Management** solution.
2. Open the Inspector canvas application in edit mode
- Locate the **Inspector Canvas App**.
  - Click on the ... **More Commands** button and select **Edit**. This will open the app in a New window.



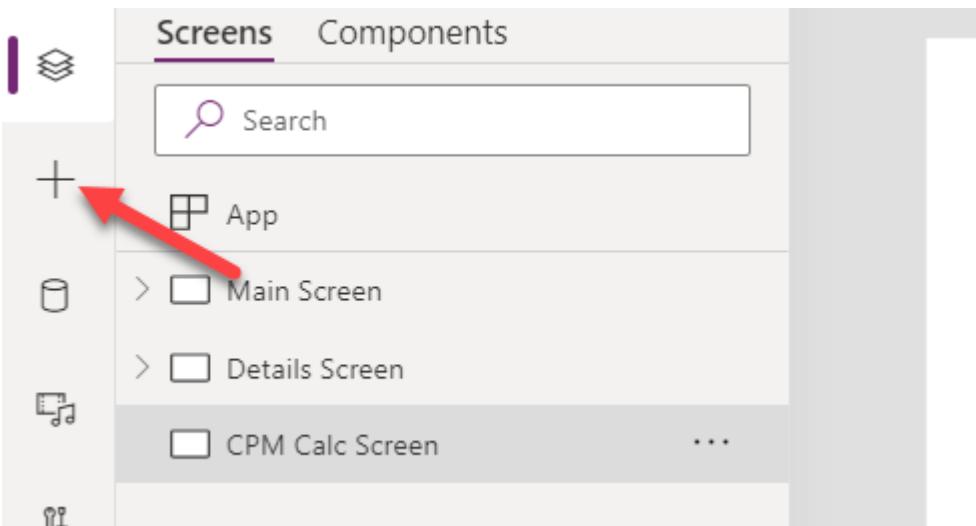
3. Add new screen to the application
- Click **New Screen** and select **Blank**.



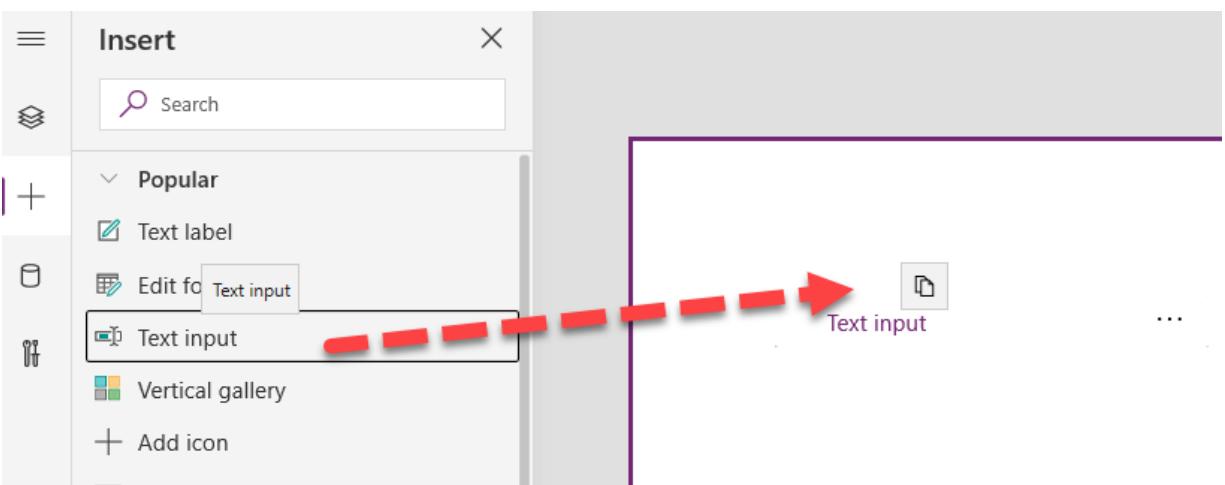
- Rename the screen **CPM Calc Screen**



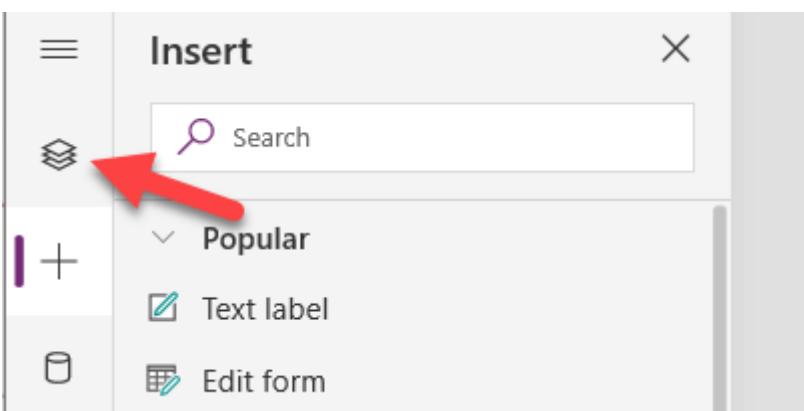
4. Add Input Text to the new screen
- Select the **CPM Calc Screen**.
  - Click **Insert**.



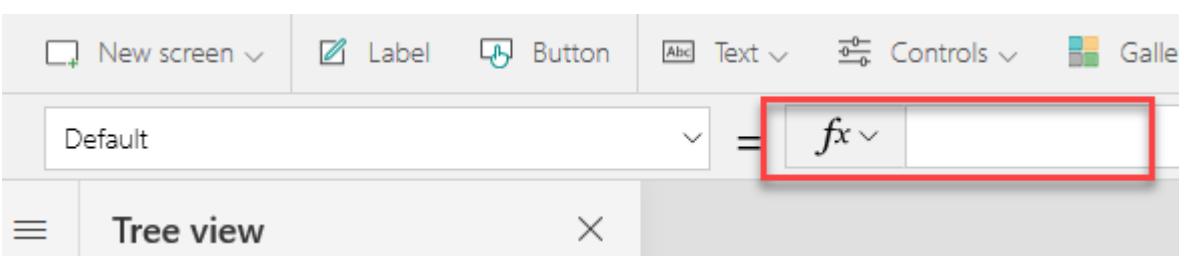
- Drag and drop **Text Input** to the screen.



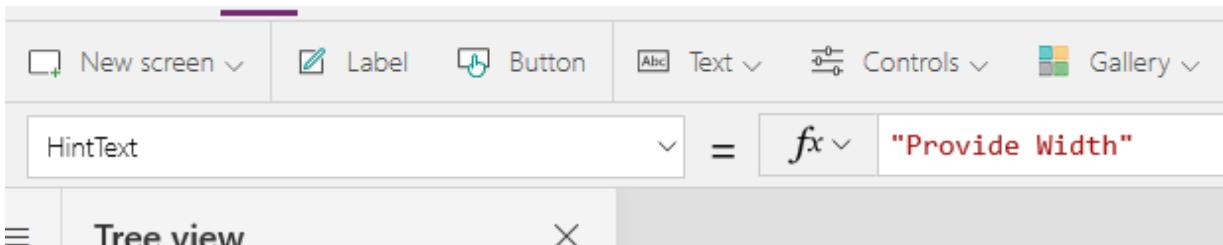
- Select the **Tree View**.



- Rename the Text Input **Width Text**.
- Remove the **Default** property of the **Width** text input.



- Change the **HintText** property of the **Width** text input to **Provide Width**.

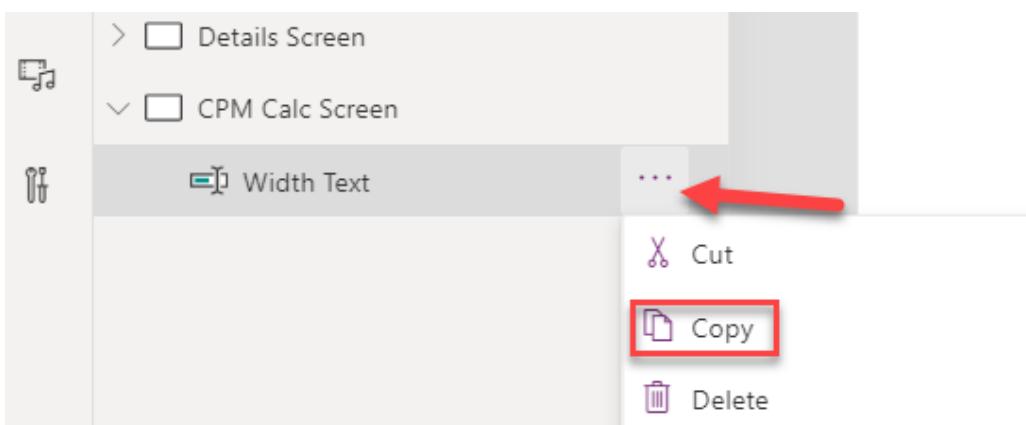


- The **Width Text** input should now look like the image below.

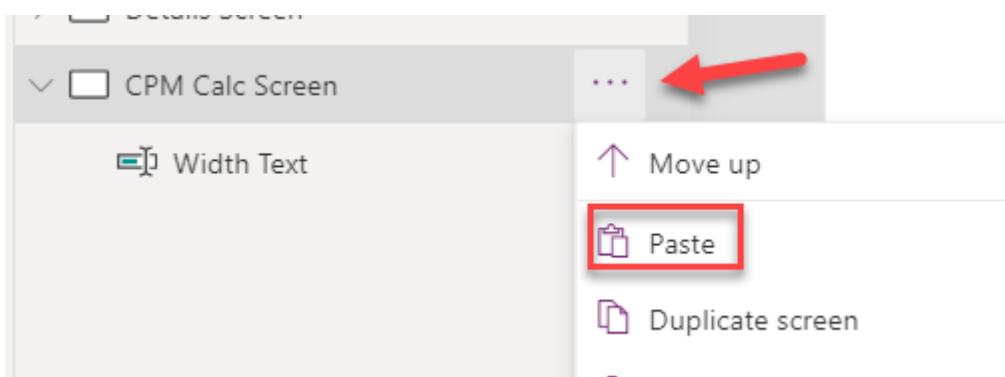


## 5. Add Height, Length, and Air Change Input Text controls

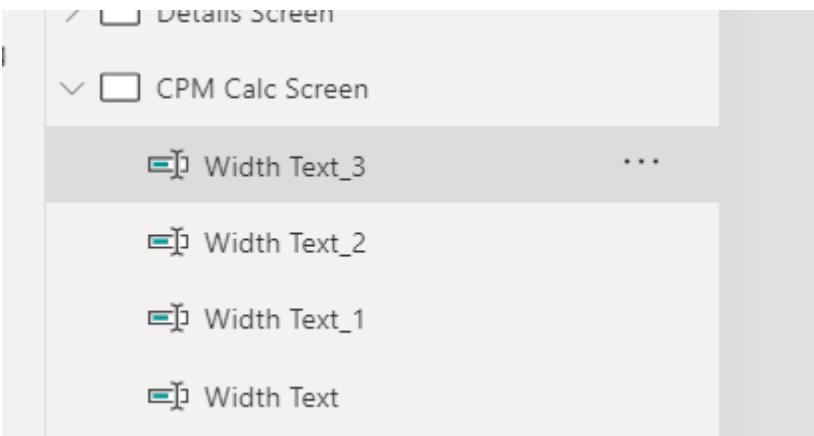
- Copy the **Width Text**.



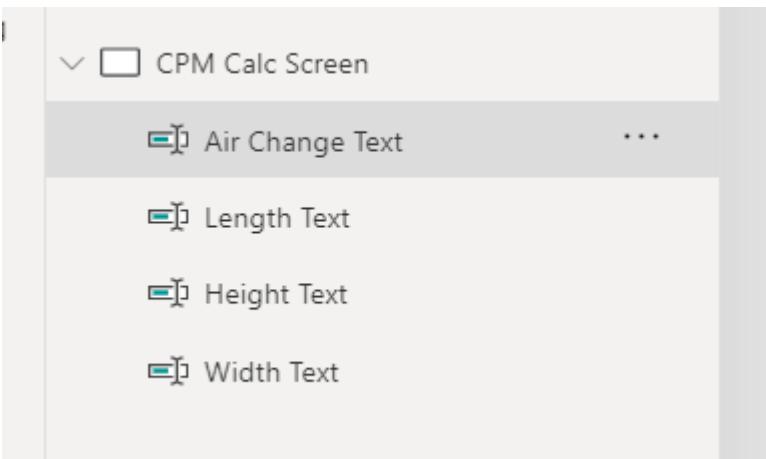
- Paste the text input you copied to the **CPM Calc Screen**.



- Paste the text input you copied to the **CPM Calc Screen** two more times.
- The **CPMCalcScreen** should now have total of four text inputs.



- Rename the input text controls **Height Text**, **Length Text**, and **Air Change Text**.



- Change the **HintText** for the three text inputs you renamed to **Provide Height**, **Provide Length**, and **Provide Air Change**, respectively.
- Resize and reposition the text inputs as shown in the image below.

Provide Width

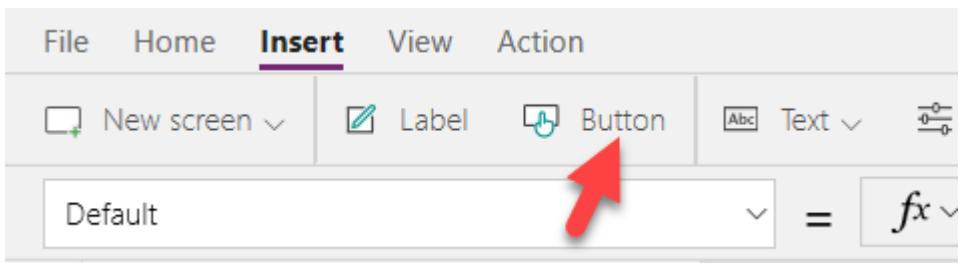
Provide Height

Provide Length

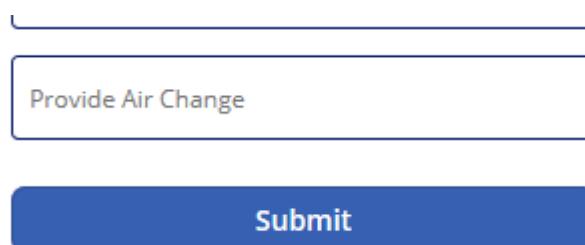
Provide Air Change

#### 6. Add button

- Go to the **Insert** tab and click Button.

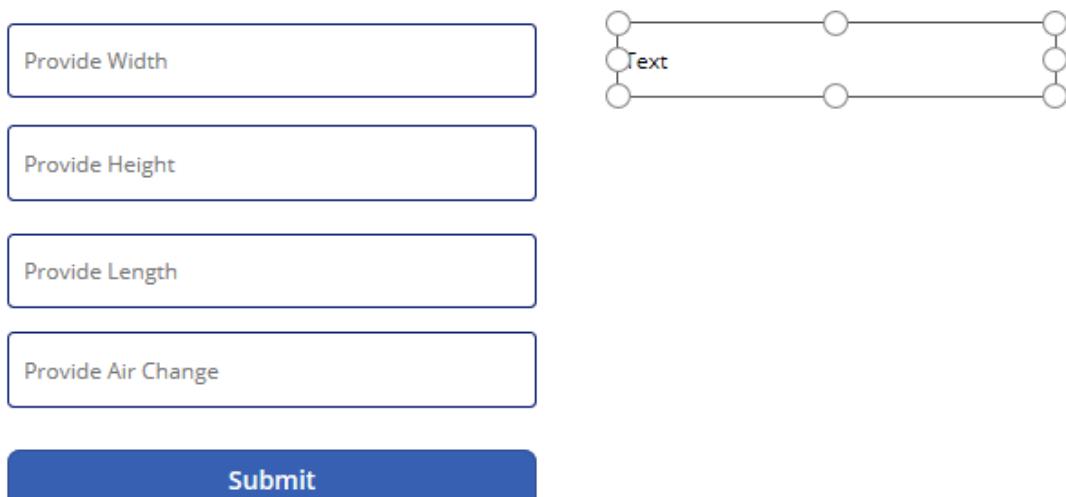


- Rename the Button **Calculate Button**.
- Change the **Calculate Button Text** value to **Submit**.
- Resize and reposition the button as shown in the image below.



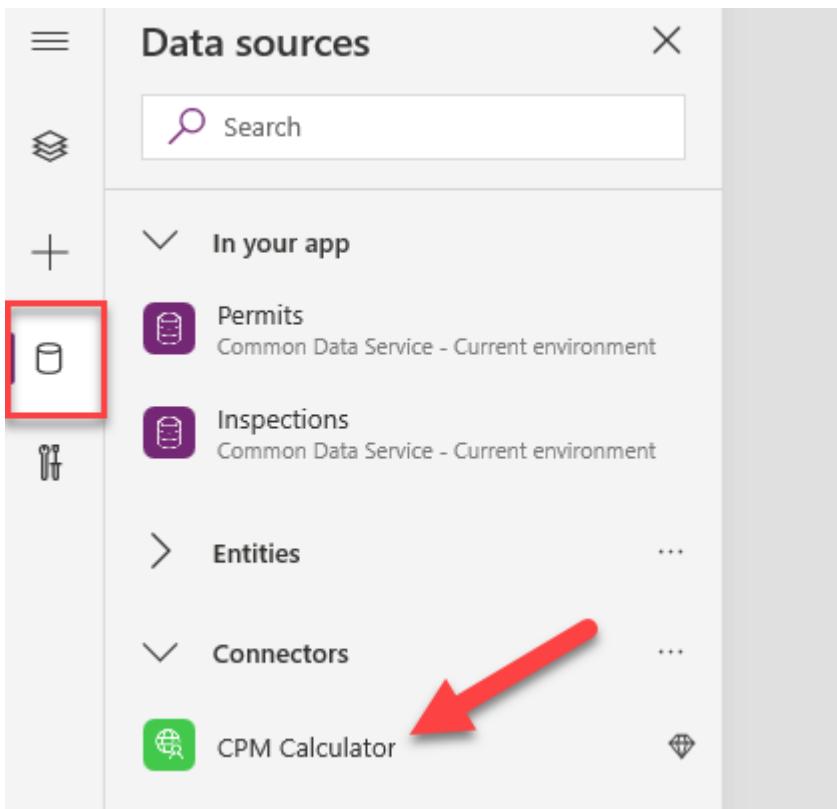
7. Add the result label to the screen

- Go to the **Insert** tab and click Label
- Rename the Label **Result Label**.
- Place the label to the right of the text inputs.

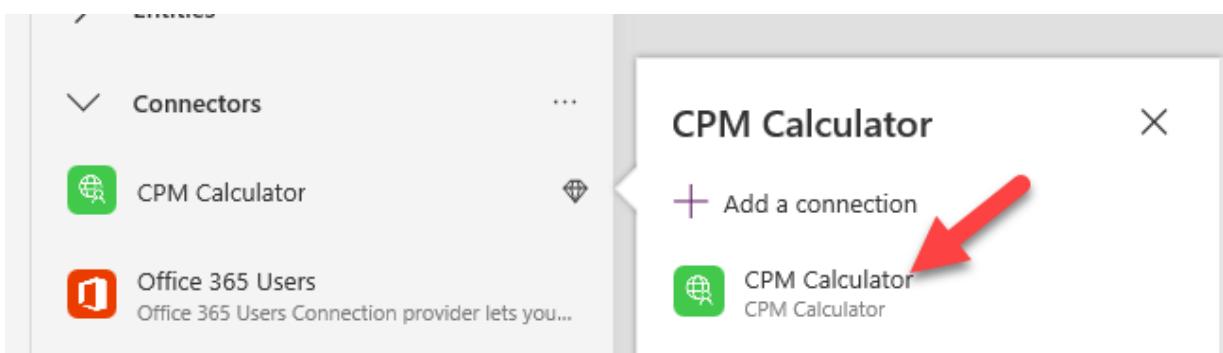


8. Add the Custom Connector to the application.

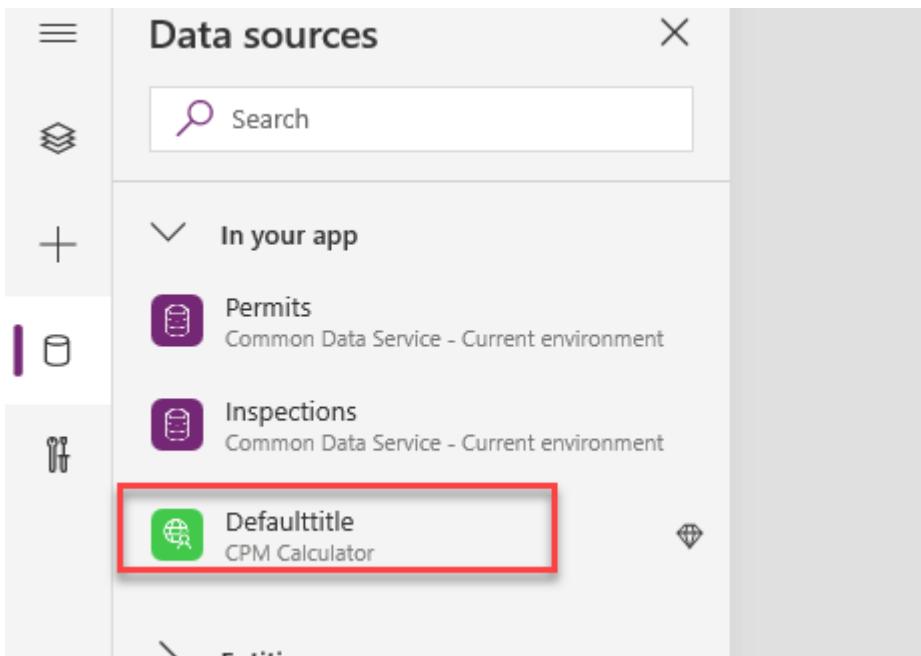
- Select the **Data Sources** tab.
- Expand **Connectors**.
- Click the **CPM Connector**.



- Click **CPM Calculator** again.



- The **CPM Calculator** should now be listed in the **In your App** section.



9. Get the calculated value when the button is clicked

- Select the **Calculate Button**.
- Set the **OnSelect** value of the **Calculate Button** to the formula below.

```
Set(CalculatedValue, Concatenate("Calculated CPM ", Text(Defaulttitle.GetRequiredCPM({Width:
```

The screenshot shows the formula bar for the 'Calculate Button'. The 'OnSelect' property is set to a formula: 'Set(CalculatedValue, Concatenate("Calculated CPM ", Text(Defaulttitle.GetRequiredCPM({Width:')). The formula bar also shows 'CalculatedValue' in the 'Text' field and 'CalculatedValue = ' in the 'CalculatedValue' field. To the right, there is a 'Tree view' pane showing the screen structure: 'Screens' tab, 'Main Screen', 'Details Screen', 'CPM Calc Screen' (selected), 'Result Label', 'Calculate Button' (selected), and 'Air Change Text'.

- Select the **Result Label** and set the **Text** value to the **CalculatedValue** variable.

The screenshot shows the formula bar for the 'Result Label'. The 'Text' property is set to 'CalculatedValue'. The formula bar also shows 'CalculatedValue = ' in the 'CalculatedValue' field. To the right, there is a 'Tree view' pane showing the screen structure: 'Screens' tab, 'Main Screen', 'Details Screen', 'CPM Calc Screen' (selected), 'Result Label' (selected), 'Calculate Button', and 'Air Change Text'.

10. Add navigation button to the Main screen

- Select the **Main Screen**.
- Go to the **Insert** tab and click **Button**.
- Rename the Button **CPM Button**.
- Change the **CPM Button Text** value to **CPM Calculator**.
- Place the button on the bottom right of the **Main Screen**.

11. Steps to navigate to the CPM Calc Screens

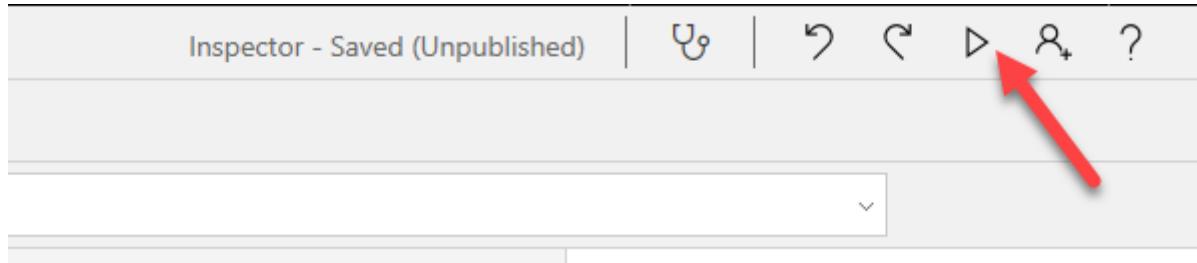
- Select the **CPM Calculator**.
- Set the **OnSelect** value of the **CPM Calculator** to the formula below.

```
Set(CalculatedValue, ""); Navigate('CPM Calc Screen', ScreenTransition.None)
```



12. Run the Application

- Select the **Main Screen** and click **Preview the App**.



- Click on **CPM Calculator** button.
- The CPM Calculator screen should load.

The form consists of four input fields stacked vertically, each with a blue border and rounded corners. The first three fields have placeholder text: 'Provide Width', 'Provide Height', and 'Provide Length'. The fourth field has placeholder text 'Provide Air Change'. Below these is a large blue rectangular button with the word 'Submit' in white capital letters.

- Provide values and click **Submit**. You can notice the loading dots on top of the screen, which confirms that the request has been initiated.

14

10

15

20

Submit

- The **Result Label** should show the calculated result from the Custom Connector.

14

Calculated CPM 700

10

15

20

Submit

- Close the Preview.
- Click **File** and **Save**.

- Publish the changes to the application.
- Click Close.

Click Done on the other window for the solution.

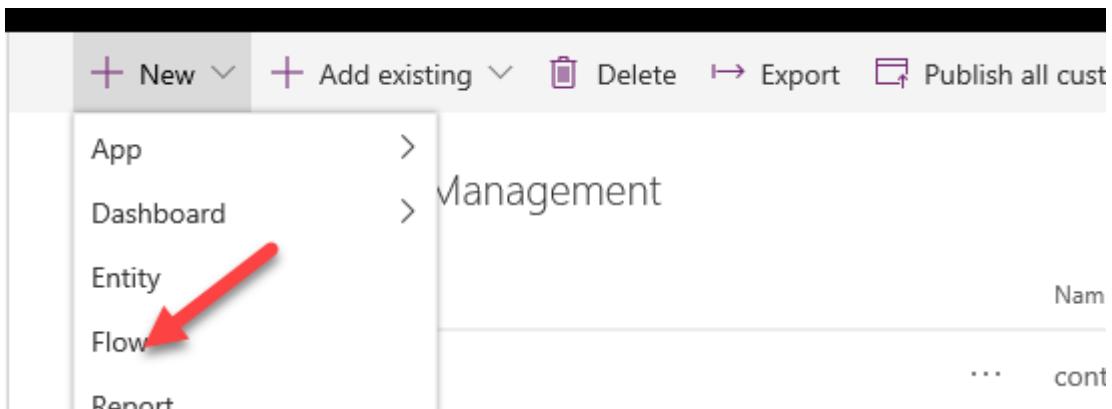
## 47.2 Task #2: Test on Flow

### 1. Open the Permit Management solution

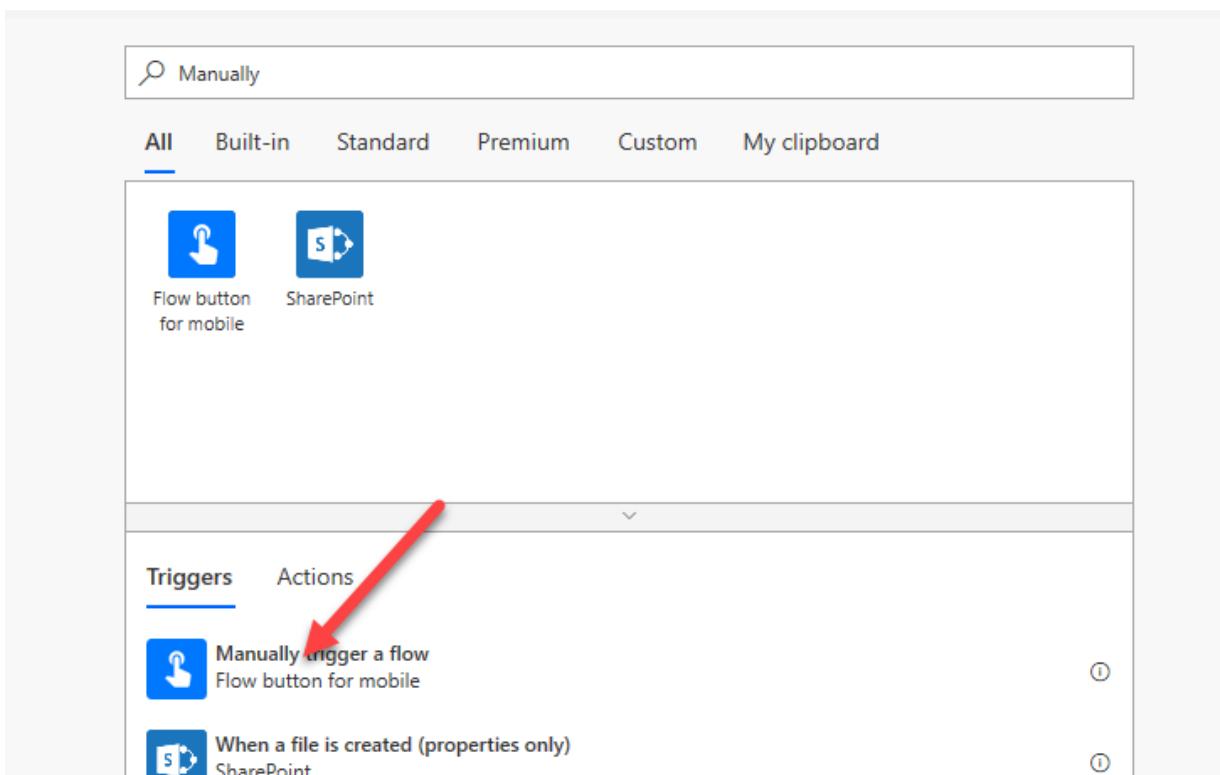
- Sign in to [Power Apps maker portal](#) and make sure you are in the **Dev** environment.
- Select **Solutions**.
- Click to open the **Permit Management** solution.

### 2. Create Flow and add trigger.

- Click **New** and select **Flow**. This will open a New window.

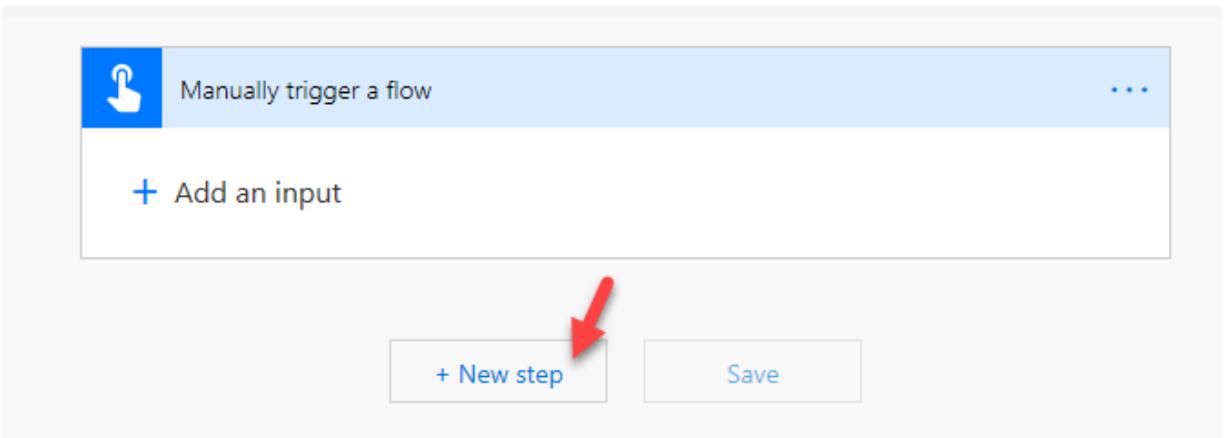


- Search for **Manually** and select **Manually Trigger Flow**.

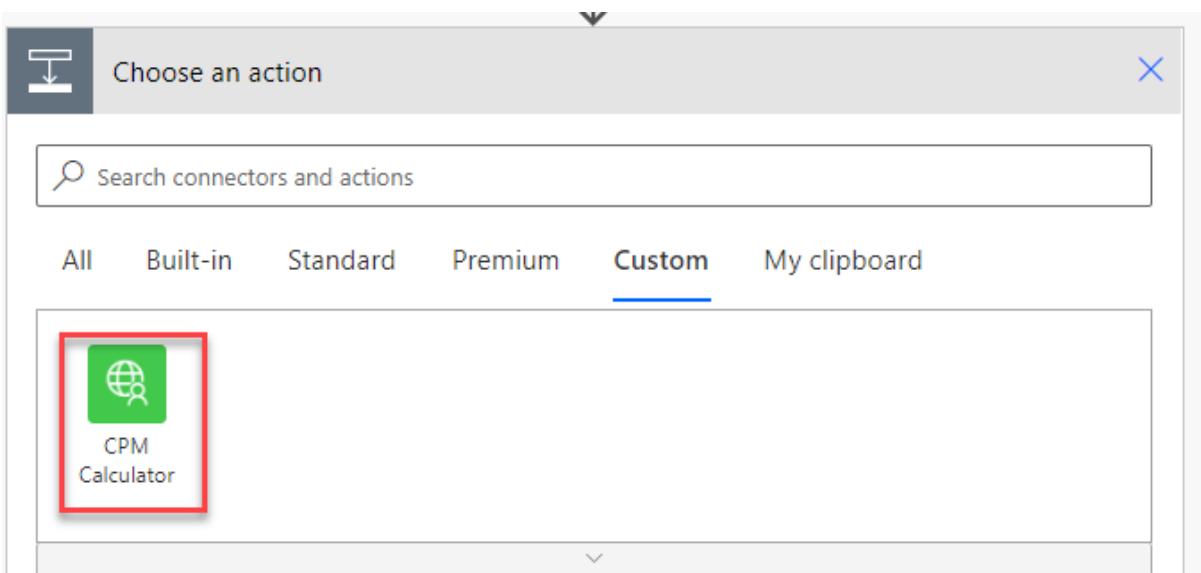


### 3. Add a step that will use the Custom Connector

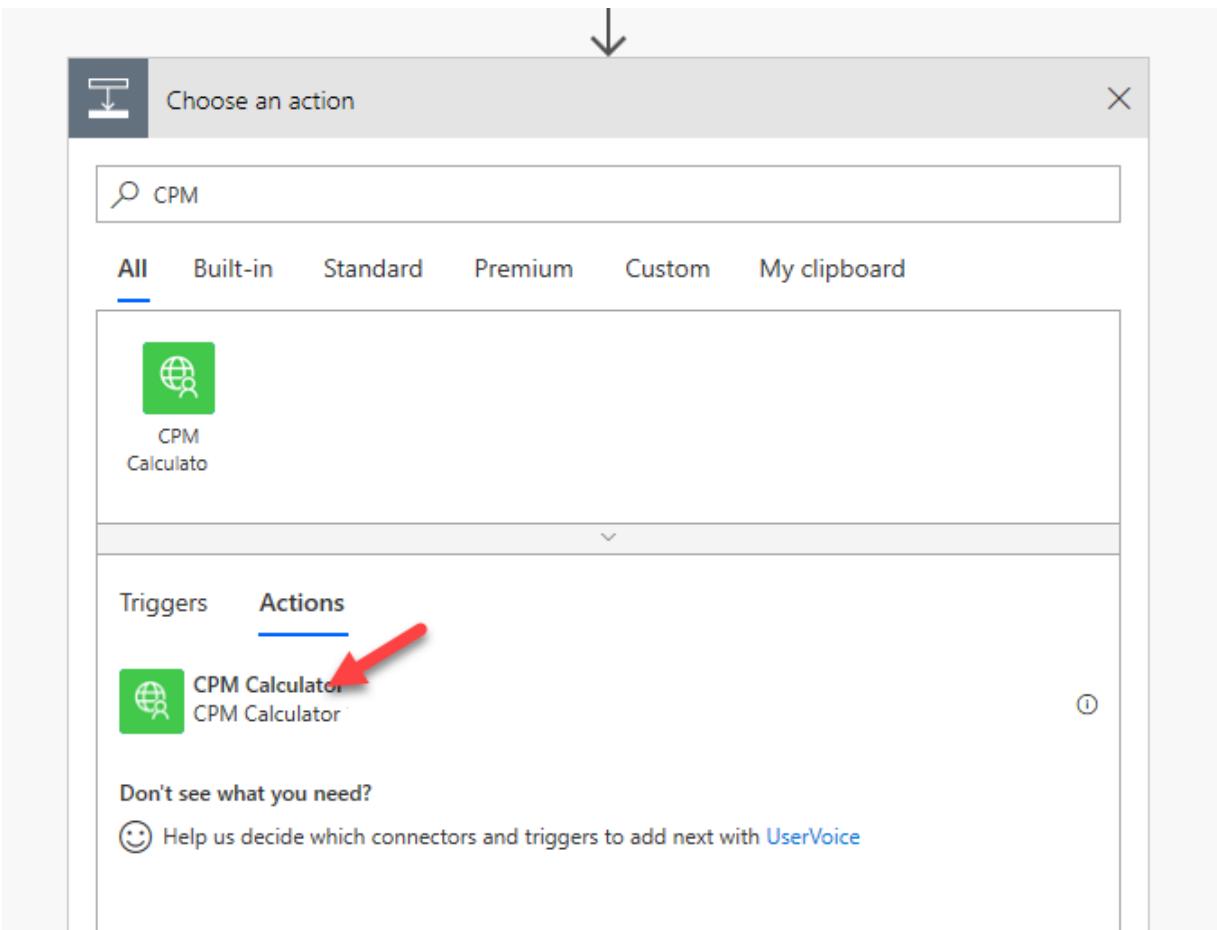
- Click **+ New Step**.



- Select the **Custom** tab and click **CPM Calculator**.



- Select **CPM Calculator** action.



4. Provide values and save

- Enter 18 for Width, 10 for Height, 18 for Length, 30 for AirChanges, and click **Save**.

Manually trigger a flow

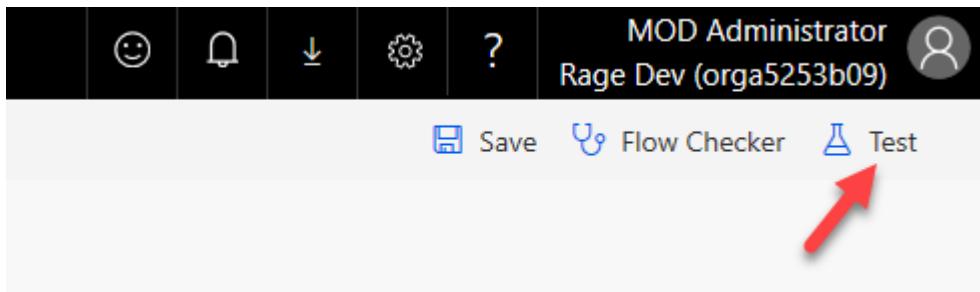
CPM Calculator

|            |    |
|------------|----|
| Width      | 18 |
| Height     | 10 |
| Length     | 18 |
| AirChanges | 30 |

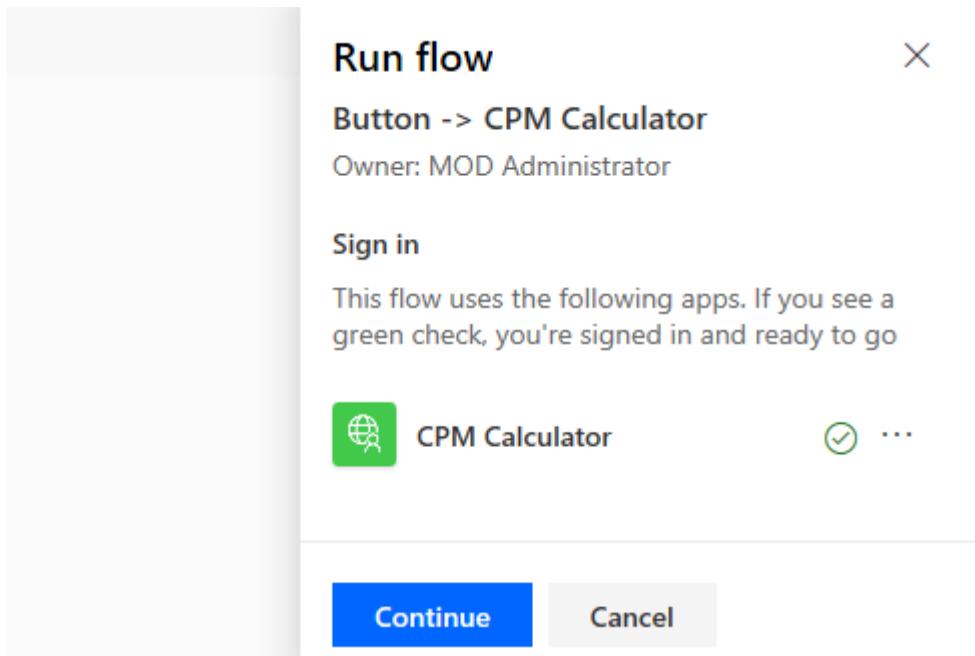
+ New step Save

5. Test the Flow

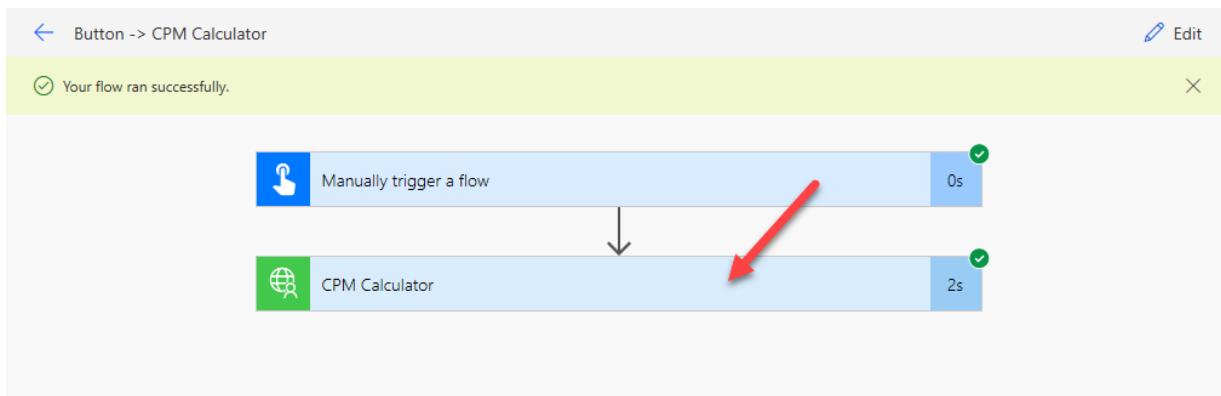
- Click Test.



- Select **I will Perform the Trigger** and click **Save & Test**.
- Click **Continue**.



- Click **Run Flow**.
- Click **Done**. The Flow should run successfully. In the Flow run history, expand the CPM Calculator action.



- You should be able to see the calculated result from the custom connector in the output of the action.

The screenshot shows a Power Automate app window titled "CPM Calculator". The top bar includes a globe icon, the title "CPM Calculator", and a "1s" timer. The main area is divided into two sections: "INPUTS" and "OUTPUTS".

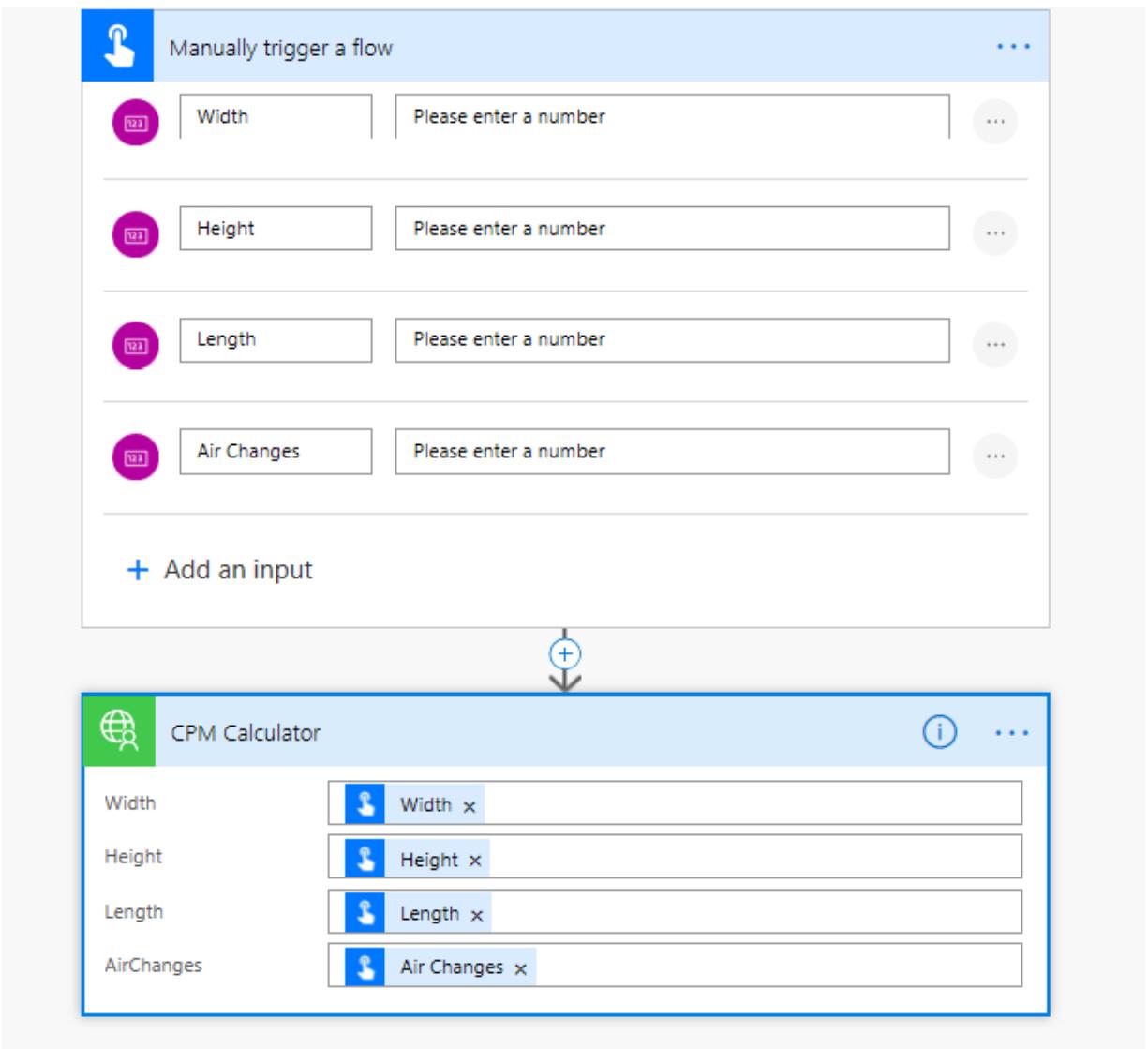
**INPUTS:**

- Width: 18
- Height: 10
- Length: 18
- AirChanges: 30

**OUTPUTS:**

- cpm: 1620

- Close the window and go back to Solution window. Click **Done**.
- Click **Publish all Customizations**.
- If you finish early, try adding input values to the Manual Button trigger for the room dimensions and use those to call the custom connector. You could also use the notification connector to send the user the required CPM. Finally, if you want to test this in a real device install the Power Automate mobile application.



### 47.3 lab: title: 'Lab 10: Application Lifecycle Management'

[!NOTE] Effective November 2020:

- Common Data Service has been renamed to Microsoft Dataverse. [Learn more](#)
- Some terminology in Microsoft Dataverse has been updated. For example, *Table* is now *table* and *field* is now *column*. [Learn more](#)

This content will be updated soon to reflect the latest terminology.

### 47.4 Lab 10 – Application Lifecycle Management

## 48 Scenario

A regional building department issues and tracks permits for new buildings and updates the remodeling of existing buildings. Throughout this course you will build applications and automation to enable the regional building department to manage the permitting process. This will be an end-to-end solution which will help you understand the overall process flow.

In this lab you will use Azure DevOps for source control of your solution assets. As you have been building your app you have been tracking all the changes in a Permit Management solution. You have exported this solution, so you had a back up copy. You have also manually imported the managed version of the solution into your production environment. As part of this lab you will see how you can automate working with solutions and use the Power Apps Azure DevOps tasks to check the changes into an Azure DevOps Repository. This

is the start of an overall ALM process that you would put in place to automate the complete lifecycle from development to production using Azure DevOps automation. In this lab you will be completing the first phase of that automation.

## 49 High-level lab steps

As part of configuring Azure DevOps ALM automation, you will complete the following

- Sign in to your Azure DevOps account
- Create an Azure DevOps project
- Configure the Power Apps ALM tasks
- Build an export solution pipeline
- Test the export from dev to Azure DevOps

### 49.1 Things to consider before you begin

- How often are you planning to run the build process?
- Is it going to be fully automated or run manually?
- How many users will be committing changes into the repositories and how often?
- How many instances are you planning to control?
- Are there any other build tasks you should consider?

## 50 Exercise #1: Initialize Azure DevOps

**Objective:** In this exercise, you will use your Azure DevOps account to track the solution assets of the Permit Management app.

### 50.1 Task #1: Sign in for Azure DevOps

1. Sign in for Azure DevOps
  - Navigate to [Azure DevOps](#)
  - Provide your admin credentials and sign in.
2. Create the Azure DevOps project
  - Enter **Permit Management** for **Project Name**, select **Private**, and click **Create Project**.

## Create a project to get started

Project name \*

Permit Management

Visibility



Public

Anyone on the internet can view the project. Certain features like TFVC are not supported.



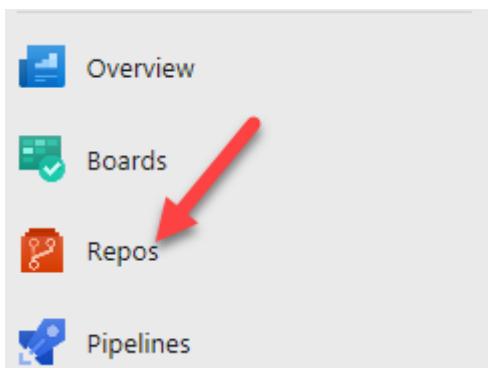
Private

Only people you give access to will be able to view this project.

+ Create project

### 3. Initialize Repository

- Select **Repos**.



- Scroll down to the bottom, check the **Add a README** checkbox, and click **Initialize**.

### Initialize master branch with a README or .gitignore

Add a README

Add a .gitignore: None ▾

Initialize

## 50.2 Task #2: Configure Power Apps ALM Tasks

### 1. Get Power Apps BuildTools

- Sign in to [Visual Studio marketplace](#)
- Search for **Power Platform Build Tools**.
- Select **Power Platform Build Tools**.

## Power Platform Build Tools

690 Results

Showing: All categories



**Power Platform Build**  
Microsoft  2K  
Automate common build and deployment tasks related to Power Platform  
 **FREE**



**Power DevOps Tools**  
Wael Hamze  8.3K  
Tasks for automating Build & Deployment of Dynamics 365 CE/CDS/PowerApps Solutions  
 **FREE**



**Power BI Build Tools**  
Bob Guidinger  
Build and Release  
Power BI  


- Click **Get it Free**.



**Power Platform Build Tools (1.0.8)**  
Microsoft |  2,050 installs |  (3) | Free  
Automate common build and deployment tasks related to Power Platform  
[Get it free](#)

- Select the **Azure DevOps** organization you created and click **Install**.

### Select an Azure DevOps organization

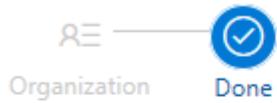
HR-PermitManagement

**Install**

For Azure DevOps Server

**Download**

- Click **Proceed to Organization**.



You are all set!

[Proceed to organization](#) [Go to Marketplace](#)

## 2. Go to Git repositories security

- Click to open the **Permit Management** project you created.

A screenshot of the Azure DevOps interface. On the left, there's a sidebar with a green 'H' icon and the text 'HR-PermitManagement'. Below it are links for 'New organization' and 'Project settings'. The main area is titled 'HR-PermitManagement' with tabs for 'Projects', 'My work items', and 'My pull requests'. A project card for 'Permit Management' is displayed, featuring an orange 'PM' icon. A red arrow points from the bottom-left towards this project card.

- Click **Project Settings**.

A screenshot of the 'Project settings' interface. On the left, there's a sidebar with icons for 'Test Plans', 'Artifacts', and 'Project settings'. A red arrow points from the bottom-left towards the 'Project settings' icon. The main area shows a 'Repos' section with a red box around the 'Repositories' link. Other options like 'Policies' and 'Test' are also visible.

- Select **Repositories**.

A screenshot of the 'Repositories' section. It shows a sidebar with 'Repos' selected and a red box around the 'Repositories' link. Other options like 'Policies' and 'Test' are listed below.

## 3. Add permissions to allow the build account to check in solution assets

- Select the **Permit Management** project.

# All Repositories

Repositories Settings Policies Permissions

## Permit Management



- Select the **Permissions** tab.
- Search for **Project Collection Build Service** and select the one without the accounts **Project Collection**.

The screenshot shows a list of service accounts under 'Project Collection Build Service Accounts'. One account, 'Project Collection Build Service (P6cc088a8-e712-472c-6cc088a8-e712-472c-b674-0c3aea505785)', is highlighted with a red box. To its right, there are four columns: '[HR-PermitManagement]', 'Bypass |', 'Bypass |', and 'Contribute'.

#### 4. Set Contribute permission for the Service Accounts

- Select the **Project Collection Build Service**.
- Locate the **Contribute** permission and select **Allow**.

#### Security for all Git repositories

The screenshot shows the security settings for 'Project Collection Build Service (Pc3ec9dbe-b645-46d1-8522-20d0bbf45c07)'. Under the 'Users' section, 'Project Collection Build Service (Pc3ec9dbe-b645-46d1-8522-20d0bbf45c07)' is selected and highlighted with a red box. In the 'Contribute' row, a dropdown menu is open with 'Allow' selected, also highlighted with a red box. Other permissions listed include 'Bypass policies when completing pull requests', 'Bypass policies when pushing', 'Contribute to pull requests', and 'Create branch', each with a 'Not set' dropdown.

- Click **Show More** to expand the menu.



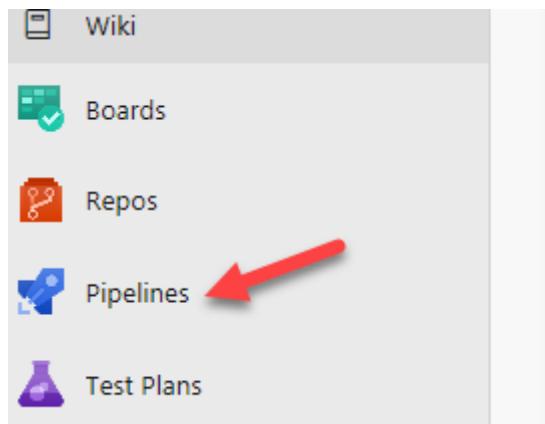
## 51 Exercise #2: Build Export Pipeline

**Objective:** In this exercise, you will build an Azure DevOps pipeline that will export the solution from the development Microsoft Dataverse environment, unpack the solution file to individual files and then check those files into the repository.

### 51.1 Task #1: Export the Solution

1. Create Build Pipeline

- Click to expand **Pipelines**.



- Click **New Pipeline**.

## Create your first Pipeline

Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.



- Click **Use the Classic Editor**.



[Use the classic editor to create a pipeline without YAML.](#)

- Don't change the default values and click **Continue**.

Select a source

Azure Repos Git GitHub GitHub Enterprise Server Subversion Bitbucket Cloud Other Git

Team project

Permit Management

Repository

Permit Management

Default branch for manual and scheduled builds

master

Continue

- Select **Empty Job**.

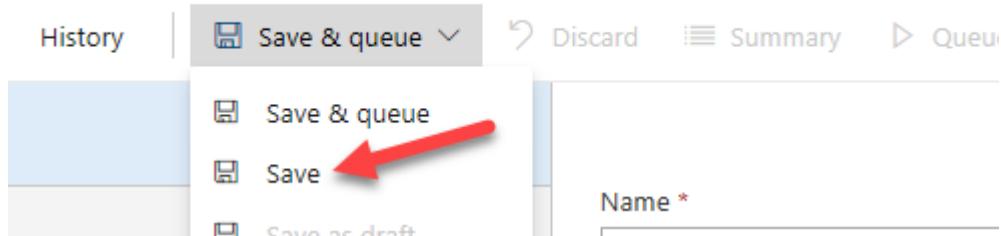
### Select a template

Or start with an [Empty job](#)



Configuration as code

- Click **Save and Queue** and select **Save**.



- Click **Save**.

## Save build pipeline

Select folder \*

...

Comment

Save

Cancel

### 2. Add Power Apps Tool Installer task

Note: The Power Apps Tool Installer needs to be run before any other Power Apps ALM tasks.

- Click + icon to add Task to **Agent Job 1**.

The screenshot shows the 'Tasks' tab of the Pipeline editor. At the top, there are tabs for Tasks, Variables, Triggers, Options, Retention, and History. Below the tabs, there's a header for 'Pipeline' and 'Build pipeline'. Underneath, there's a section for 'Get sources' and another for 'Agent job 1'. In the 'Agent job 1' section, there's a 'Run on agent' button. To the right of this section, there's a large blue '+' icon with a red arrow pointing to it, indicating where to click to add a new task.

- Search for **Power Platform Tool**, hover over select **Power Platform Tool Installer** and click **Add**.

The screenshot shows the 'Add tasks' search results for 'power platform tool'. A search bar at the top contains the text 'power platform tool'. Below the search bar, there's a list of tasks. One task, 'Power Platform Tool Installer', is highlighted with a purple icon and has an 'Add' button to its right. There's also a 'Marketplace' link at the bottom left.

### 3. Add PowerApps Export Solution task

- Search for **Export**.
- Hover over **Power Platform Export Solution** and click **Add**.

Add tasks | Refresh

export

X

The screenshot shows the 'Power Platform Export Solution' task within a larger workflow. The task icon is a purple square with a white gear and wrench. The task name is 'Power Platform Export Solution'. Below it, the sub-task 'Power Platform Export Solution' is listed. A blue 'Add' button is located in the bottom right corner of the task area.

4. Open PowerApps Export Solution

- Select the **Power Platform Export Solution** task.

The screenshot shows the 'Power Platform Import Solution' task in the Power Automate interface. The task icon is a purple square with a white gear and wrench. The task name is 'Power Platform Import Solution'. A red callout box highlights the status message 'Some settings need attention' in red text. A blue checkmark icon is to the right of the task name. A blue '+' button is located in the top right corner of the task area.

5. Get your Dev Environment URL

- Start a new browser window or tab and sign in to [Power Platform admin center](#)
- Select **Environments** and click to open the **Dev** environment.
- Copy the **Environment URL** and keep it in your clipboard.

The screenshot shows the 'Details' view of a Dev environment in the Power Platform Admin center. The environment URL is 'orga5253b09.crm.dynamics.com'. The status is 'Ready' and the region is 'North America'. A context menu is open over the URL field, with the 'Copy link' option highlighted by a red arrow. Other options in the menu include 'Open in new tab', 'Open in new InPrivate window', and 'Save target as'.

- Close the **Power Platform Admin** browser window or tab.

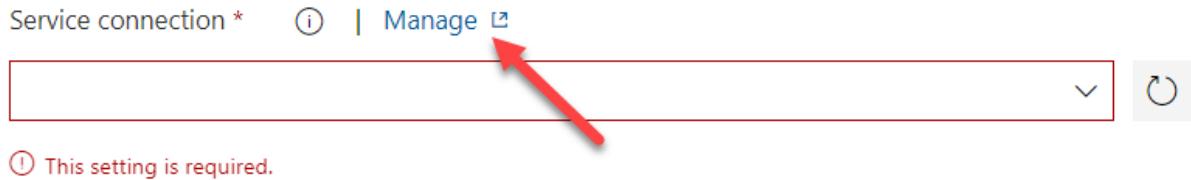
6. Create Generic Service Connection

- Go back to the **Pipeline**.
- Make sure you still have the **Power Platform Export Solution** task selected.
- Click **Manage** service connection. This will open a new window.

Service connection \* ⓘ | Manage ↗

ⓘ This setting is required.

• Click Create Service Connection.



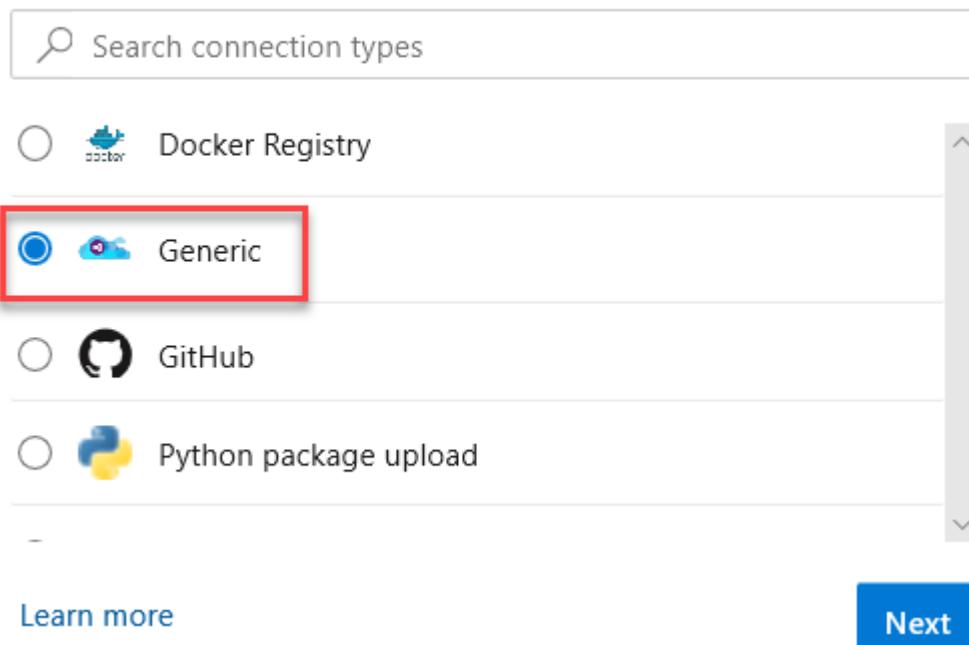
## Create your first service connection

Service connections help you manage, protect, and reuse authentications to external services.

[Create service connection](#)

- Select **Generic** and click **Next**.

Choose a service or connection type



Search connection types

- Docker Registry
- Generic
- GitHub
- Python package upload

Learn more

Next

- Paste the **Environment URL** you copied in Server URL, provide your admin credentials, provide a connection name, and click **Save**.

## New Generic service connection

X

### Server URL

https://org627a39f4.crm.dynamics.com/

### Authentication

Username (optional)

Username for connecting to the endpoint

Password (optional)

Password for connecting to the endpoint

### Details

Service connection name

Dev Connection

Description (optional)

### Security

Grant access permission to all pipelines

[Learn more](#)

Back

Save

- Close the **Service Connections** browser window or tab.

7. Select the Generic Service Connection you created as the Power Apps Environment URL

- Go back to the **Build Pipeline** tasks and make sure you still have Power Apps Export Solution task selected.
- Locate the **Power Apps Environment URL** Column and click **Refresh**.

Service connection \* [\(i\)](#) | [Manage](#)



[\(i\) This setting is required.](#)



- Select the **Generic Service Connection** you created.

Service connection \* [\(i\)](#) | [Manage](#)

Dev Connection [\(i\)](#) [Save](#)

Solution Input File \* [\(i\)](#)

- Enter `$(SolutionName)` for **Solution Name**, `$(Build.ArtifactStagingDirectory)$(SolutionName).zip` for **Solution Output File**.

Display name \* [\(i\)](#)

Power Platform Export Solution

Authentication type \* [\(i\)](#)

Username/password (no MFA support)  Service Principal/client secret (supports MFA)

Service connection \* [\(i\)](#) | [Manage](#)

Dev Connection [\(i\)](#) [Save](#)

Solution Name \* [\(i\)](#)

`$(SolutionName)`

Solution Output File \* [\(i\)](#)

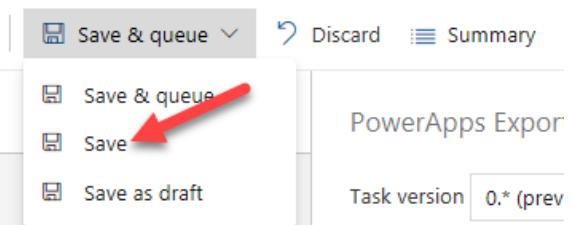
`$(Build.ArtifactStagingDirectory)\$(SolutionName).zip` [...](#)

Export as Managed Solution [\(i\)](#)

Advanced [\(i\)](#)

- Click **Save and Queue** and select **Save**.

[... > Permit Management-CI](#)



Tasks Variables Triggers Options Retention History [Save & queue](#) [Discard](#) [Summary](#)

Pipeline Build pipeline

Get sources Permit Management master

PowerApps Export Task version 0.\* (prev)

- Click **Save** again.

8. Add Unpack task. This task will take the solution zip file and expand it into a file for each solution component.

- Click **+ Add Task**.

Agent job 1  
Run on agent

Power Platform Tool Installer  
Power Platform Tool Installer

- Search for **Unpack**.
- Hover over **Power Platform Unpack Solution** and click **Add**.

Add tasks | Refresh

unpack



Power Platform Unpack Solution  
Power Platform Unpack Solution

Add

9. Provide Unpack settings information

- Select the **Unpack** task.
- Enter `$(Build.ArtifactStagingDirectory)$(SolutionName).zip` for **Solution Input File**, `$(Build.SourcesDirectory)$(SolutionName)` for **Target Folder**.

Display name \*

Power Platform Unpack Solution

Solution Input File \*

`$(Build.ArtifactStagingDirectory)\$(SolutionName).zip`

Target Folder to Unpack Solution \*

`$(Build.SourcesDirectory)\$(SolutionName)`

Type of Solution

Unmanaged

- Click **Save and Queue** and select **Save**.
- Click **Save** again.

10. Allow scripts to access the OAuth Token.

- Select **Agent Job 1**.

Pipeline

Build pipeline

Get sources

Permit Management master

Agent job 1

Run on agent

Power Platform Tool Installer

Power Platform Tool Installer

- Scroll down and check the **Allow Scripts to Access the OAuth Token** checkbox.

Select dependencies

Additional options ^

Allow scripts to access the OAuth token ⓘ

Run this job ⓘ

11. Add Command Line task

- Click **+ Add a Task**.

Permit Management master

Agent job 1

Run on agent

Power Platform Tool Installer

- Search for **Command Line**.
- Hover over **Command Line** and click **Add**.

Discard Summary Queue ...

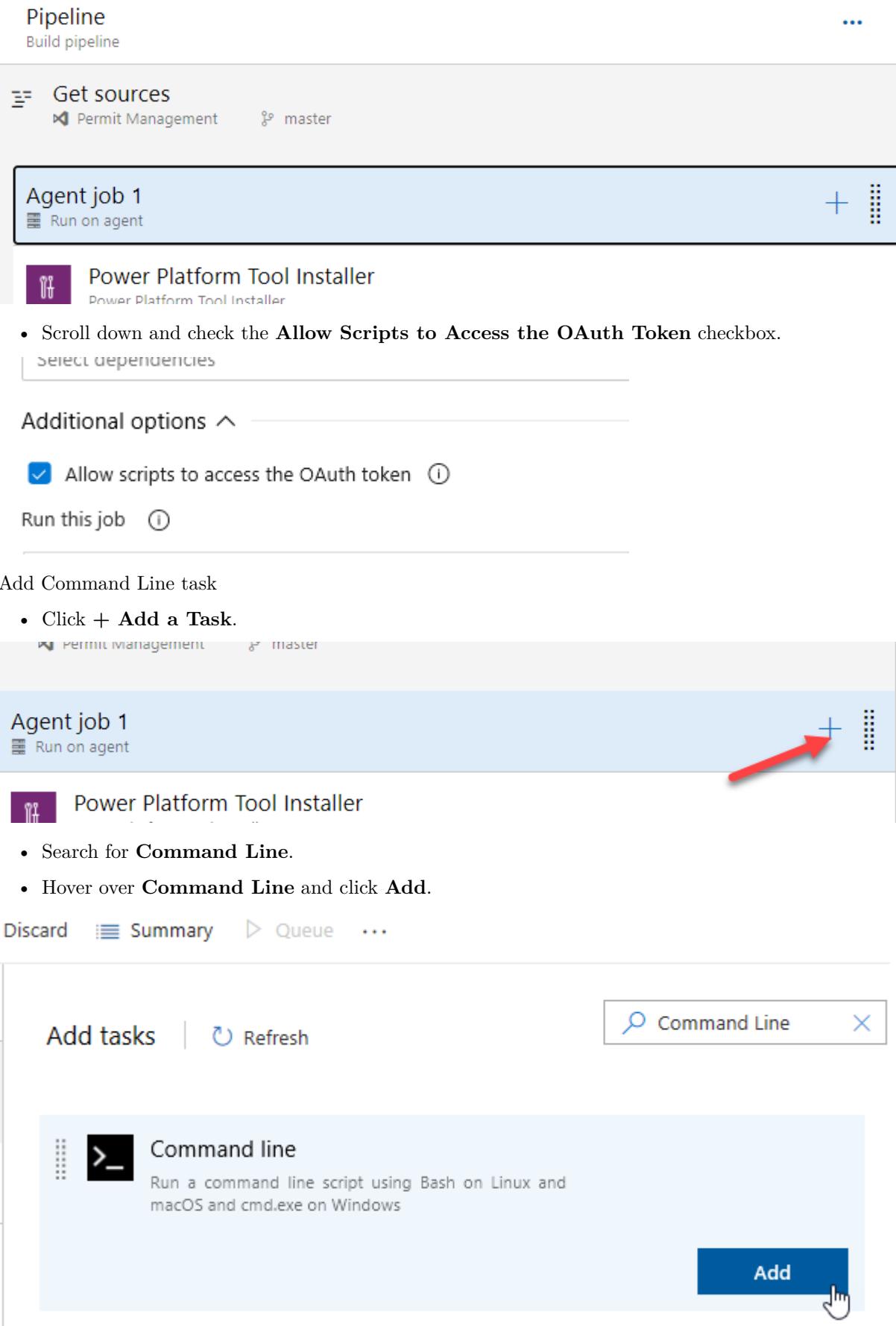
Add tasks | Refresh

Command Line

Command line

Run a command line script using Bash on Linux and macOS and cmd.exe on Windows

Add



12. Add Scripts to the Command Line task. This task will be used to check in the solution file changes to the repo.
- Select the **Command Line** task.

- Paste the script below in the **Script** text area. Replace **user@myorg.onmicrosoft.com** with your admin username.

```
echo commit all changes
git config user.email "user@myorg.onmicrosoft.com"
git config user.name "Automatic Build"
git checkout master
git add --all
git commit -m "solution init"
echo push code to new repo
git -c http.extraheader="AUTHORIZATION: bearer $(System.AccessToken)" push origin master
```

Display name \*

Command Line Script

Script \* ⓘ

```
echo commit all changes
git config user.email "admin@M365x826979.onmicrosoft.com"
git config user.name "Automatic Build"
git checkout master
git add --all
git commit -m "solution init"
echo push code to new repo
git -c http.extraheader="AUTHORIZATION: bearer
$(System.AccessToken)" push origin master
```

### 13. Add Solution Name variable

- Select the **Variables** tab.
- Click **+ Add**.

| Name ↑              | Value                            |
|---------------------|----------------------------------|
| system.collectionId | 794a909a-8e15-45b6-8f90-cc1dd641 |
| system.debug        | false                            |
| system.definitionId | 1                                |
| system.teamProject  | Permit Management                |

- Enter **SolutionName** for **Name** and **PermitManagement** for **Value**.

|                    |                                    |
|--------------------|------------------------------------|
| system.teamProject | Permit Management                  |
| SolutionName       | <input type="button" value="Add"/> |

- Click **Save and Queue** and select **Save**.
- Click **Save** again.

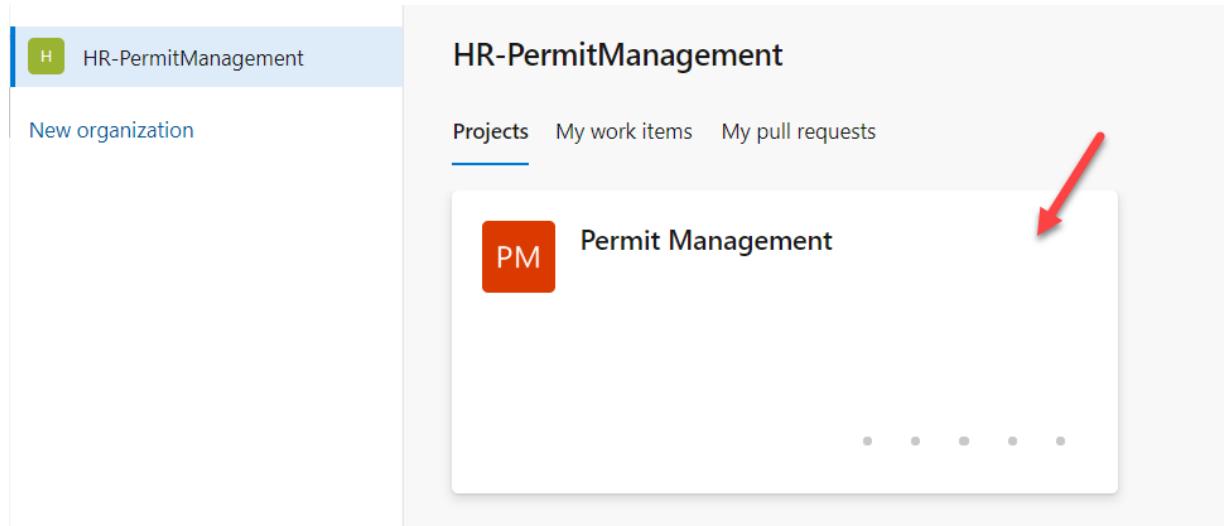
## 52 Exercise #3: Test the Pipeline

**Objective:** In this exercise, you will test the build pipeline you created.

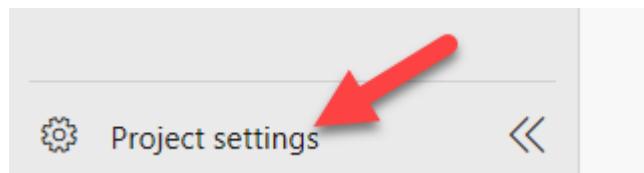
### 52.1 Task #1: Run the Pipeline

1. Open the build pipeline

- Sign in to [Azure DevOps](#) and click to open the **Permit Management** project.



- Click **Project Settings**.



- Select **Repositories**.



- Select the **Permit Management** repository.
- Select the **Permissions** tab.
- Select **Permit Management Build Service** and **Allow Contribute**.

## Permit Management

[Browse](#) [Rename](#) [Delete](#)

Settings Policies Permissions

Inheritance ⓘ

Search for users or groups

Azure DevOps Groups

Users

Permit Management Build Service (HR-PermitManagement) 

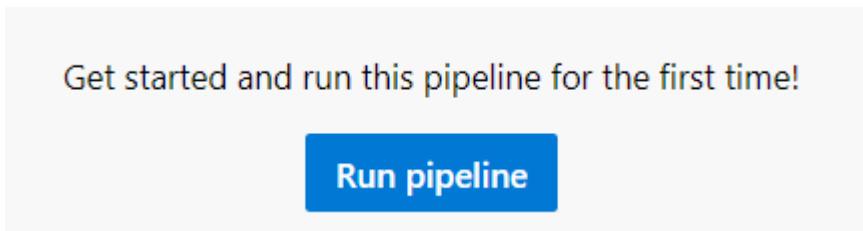
Project Collection Build Service (P6cc088a) 

|                                               | Permit Management Build Service (HR-PermitManagement)                                                 |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Bypass policies when completing pull requests | Not set                                                                                               |
| Bypass policies when pushing                  | Not set                                                                                               |
| Contribute                                    | Allow              |
| Contribute to pull requests                   | Not set                                                                                               |
| Create branch                                 | Not set                                                                                               |
| Create tag                                    | Allow (inherited)  |

- Select **Pipelines | Pipelines**.



- Select **Permit Management-CI**.
- Click **Run Pipeline**.



- Click **Run** again and wait.

Cancel **Run**

Jobs

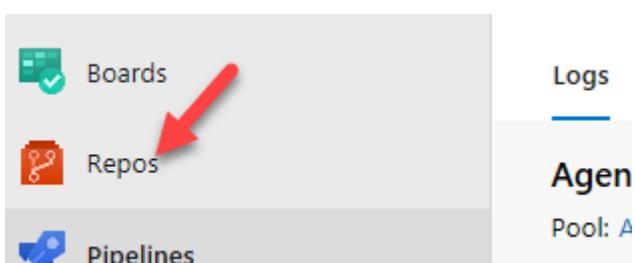
| Name                                                                                            | Status  | Duration |
|-------------------------------------------------------------------------------------------------|---------|----------|
| Agent job 1  | Success | ⌚ 3m 38s |

- The Build tasks should run and succeed

| Jobs |                           |        |
|------|---------------------------|--------|
| ▼    | Agent job 1               | 3m 38s |
| ✓    | Initialize job            | 2s     |
| ✓    | Checkout Permit Mana...   | 7s     |
| ✓    | Power Platform Tool In... | 29s    |
| ✓    | Power Platform Ex...      | 2m 48s |
| ✓    | Power Platform Unpack...  | 5s     |
| ✓    | Command Line Script       | 4s     |
| ✓    | Post-job: Checkout Pe...  | <1s    |
| ✓    | Finalize Job              | <1s    |
| ✓    | Report build status       | <1s    |

2. Review the Repository

- Select Repos.



- You should see **PermitManagement** folder. Click to open the folder.

The screenshot shows a GitHub repository interface for a 'Permit Management' project. The left sidebar lists 'Permit Management' and 'README.md'. The main area shows a table with columns for Name, Last change, and Commits. Two items are listed: 'PermitManagement' (last changed 3 minutes ago, commit d6fc2029) and 'README.md' (last changed 18 hours ago, commit 5ed60ceb). A red arrow points to the 'PermitManagement' row.

- The content of the folder should look like the image below.

| Name ↑                                                | Last change   | Commits                                |
|-------------------------------------------------------|---------------|----------------------------------------|
| AppModules/contoso_PermitManagement                   | 4 minutes ago | d6fc2029 solution init Automatic Build |
| AppModuleSiteMaps/contoso_PermitManagement            | 4 minutes ago | d6fc2029 solution init Automatic Build |
| CanvasApps                                            | 4 minutes ago | d6fc2029 solution init Automatic Build |
| Controls/contoso_contoso.timelinecontrol              | 4 minutes ago | d6fc2029 solution init Automatic Build |
| Entities                                              | 4 minutes ago | d6fc2029 solution init Automatic Build |
| OptionSets                                            | 4 minutes ago | d6fc2029 solution init Automatic Build |
| Other                                                 | 4 minutes ago | d6fc2029 solution init Automatic Build |
| PluginAssemblies/ContosoPackageProject-1B4AED25-0B... | 4 minutes ago | d6fc2029 solution init Automatic Build |
| SdkMessageProcessingSteps                             | 4 minutes ago | d6fc2029 solution init Automatic Build |
| WebResources                                          | 4 minutes ago | d6fc2029 solution init Automatic Build |
| Workflows                                             | 4 minutes ago | d6fc2029 solution init Automatic Build |

You may examine the content of each folder.

## 52.2 Task #2: Modify Solution

- Open the Permit Management solution

- Sign in to [Power apps maker portal](#) and make sure you have the **Dev** environment selected.
- Select **Solutions**.
- Click to open the **Permit Management** solution.

The screenshot shows the 'Power Apps Checker base' interface. On the left, there's a sidebar with 'Flows', 'AI Builder (preview)', and 'Solutions'. The main area lists solutions: 'PowerApps Checker Update', 'PowerApps Checker', and 'Permit Management'. A red arrow points to the 'Permit Management' solution.

- Open the Permit Table form for edit

- Click to open **Permit Table**.

|                     |                                  |              |  |
|---------------------|----------------------------------|--------------|--|
| Inspection Type     | ... contoso_inspectiontype       | Option Set   |  |
| Inspector           | ... contoso_inspector_cd9b1      | Canvas App   |  |
| Permit              | ... contoso_permit               | Entity       |  |
| Permit Form Scripts | ... contoso_PermitFormScripts.js | Web Resource |  |

- Select the **Forms** tab and click to open the **Main** form.

Solutions > Permit Management > Permit

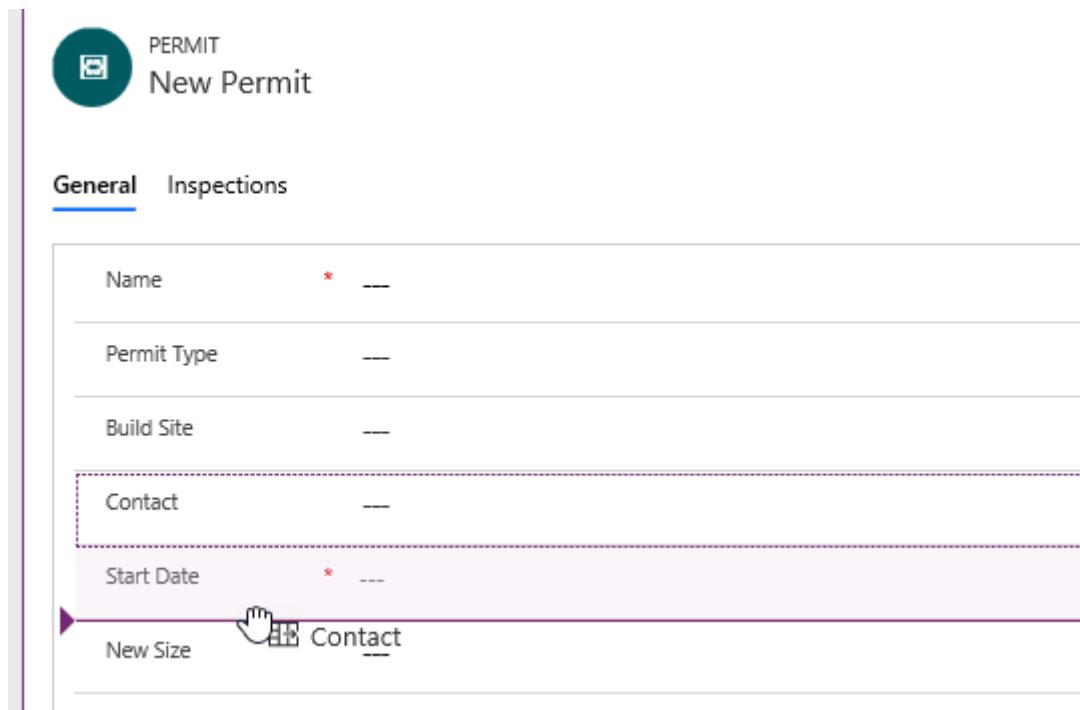
Fields   Relationships   Business rules   Views   **Forms**   Dashboards   Charts   Keys   Data

Model-driven

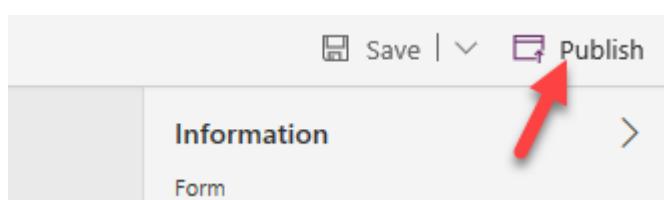
| Name ↑ ↓    | Form type           |
|-------------|---------------------|
| Information | ...  Main           |
| Information | ... Quick View Form |
| Information | ... Card            |

- Move the Contact lookup Column

- Drag the **Contact** lookup Column and drop it between the **Start Date** and **New Size** Columns.



- Click **Save**.
- Click **Publish** and wait for the publishing to complete.



### 52.3 Task #3: Run Build Pipeline

1. Open Permit Management DevOps project

- Sign in to [Azure DevOps](#)
- Click to open the **Permit Management** project

The screenshot shows the Azure DevOps interface for the 'HR-PermitManagement' organization. On the left, there's a navigation bar with 'HR-PermitManagement' selected. Below it, there are links for 'New organization'. In the center, there's a 'Projects' tab with several cards. One card is highlighted in orange and labeled 'PM' with the text 'Permit Management'. To the right of the card, a red arrow points towards it.

2. Run the build pipeline again

- Select **Pipelines**.

The screenshot shows the 'Pipelines' section in the Azure DevOps interface. On the left, there are three icons: 'Repos', 'Pipelines' (which has a red arrow pointing to it), and 'Test Plans'. To the right, there's a 'Help' section with a 'Description' field and a '+' button.

- Select **Permit Management-CI**.
- Click **Run Pipeline**.

The screenshot shows the details of the 'Permit Management-CI' pipeline. At the top, there's a back arrow, the pipeline name, 'Edit' and 'Run pipeline' buttons (the latter has a red arrow pointing to it), and a more options menu. Below that, there are tabs for 'Runs', 'Branches', and 'Analytics'. Under the 'Runs' tab, there's a table with columns for 'Description' and 'Stages'. The first run is listed with a green checkmark next to its name.

- Click **Run** and wait for the run to complete.
- Open the job after it completes.

#### Jobs

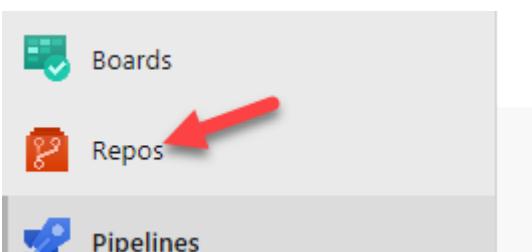
| Name        | Status  | Duration |
|-------------|---------|----------|
| Agent job 1 | Success | 2m 50s   |

- All the tasks should succeed again.

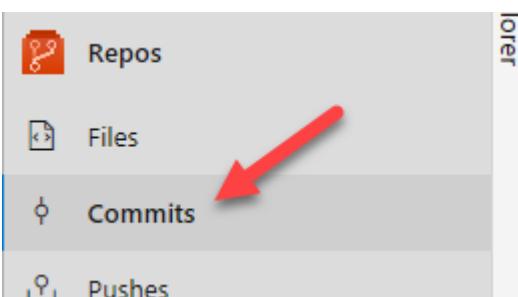
| Jobs                      |        |  |
|---------------------------|--------|--|
| Agent job 1               | 2m 50s |  |
| Initialize job            | 2s     |  |
| Checkout Permit Mana...   | 4s     |  |
| Power Platform Tool In... | 20s    |  |
| Power Platform Ex...      | 2m 16s |  |
| Power Platform Unpack...  | 4s     |  |
| Command Line Script       | 1s     |  |
| Post-job: Checkout Pe...  | <1s    |  |
| Finalize Job              | <1s    |  |
| Report build status       | <1s    |  |

3. Check the Repository for the new changes

- Select **Repos**.



- Select **Commits**.



- Click to open then topmost commit.

## Commits

| Graph | Commit                                                                  |
|-------|-------------------------------------------------------------------------|
|       | <b>solution init</b><br>428adf42  Automatic Build Today at 12:58 PM     |
|       | <b>solution init</b><br>546fbe50  Automatic Build Today at 12:41 PM     |
|       | <b>Added README.md</b><br>1aa9d981  MOD Administrator Today at 10:10 AM |

- The changeset should look like the image below.

Showing 1 file change: 1 edit

✓ </> [\(d6b00445-9f8f-41fe-90ec-97ee85b83448\).xml](#) +6 -6  
/PermitManagement/Entities/contoso\_Permit/FormXml/main/(d6b00445-9f8f-41fe-90ec-97ee85b83448).xml

```
... ...
44 44 </cell>
45 45 </row>
46 46 <row>
47 - <cell id="{ad3fe2d7-9bba-4418-9b42-12d71da8e89d}" showlabel="true" locklevel="0">
47 + <cell id="{772912b4-f1be-4669-aeb4-16c0d20ec91f}" showlabel="true" locklevel="0">
48 48 <labels>
49 - <label description="Contact" languagecode="1033" />
49 + <label description="Start Date" languagecode="1033" />
50 50 </labels>
51 - <control disabled="false" id="contoso_contact" classid="{27080D3B-09AF-4782-9025-509E298DE0A}" datafieldname="contoso_contact" />
51 + <control disabled="false" id="contoso_startdate" classid="{5B773807-9FB2-42DB-97C3-7A91EFF8ADF}" datafieldname="contoso_startdate" />
52 52 </cell>
53 53 </row>
54 54 <row>
55 - <cell id="{772912b4-f1be-4669-aeb4-16c0d20ec91f}" showlabel="true" locklevel="0">
55 + <cell id="{ad3fe2d7-9bba-4418-9b42-12d71da8e89d}" showlabel="true" locklevel="0">
56 56 <labels>
57 - <label description="Start Date" languagecode="1033" />
57 + <label description="Contact" languagecode="1033" />
58 58 </labels>
59 - <control disabled="false" id="contoso_startdate" classid="{5B773807-9FB2-42DB-97C3-7A91EFF8ADF}" datafieldname="contoso_startdate" />
59 + <control disabled="false" id="contoso_contact" classid="{27080D3B-09AF-4782-9025-509E298DE0A}" datafieldname="contoso_contact" />
60 60 </cell>
61 61 </row>
62 62 <row>
```

#### 4. View side-by-side.

- Click **View**.

{c2253a57-fbea-47bc-9032-b68a47655e01}.xml -7 +7  
/PermitManagement/Entities/contoso\_Permit/FormXml/main/{c2253a57...}

View

---

```

44 44 </cell>
45 45 </row>
46 46 <row>
47 - <cell id="{1ea0b6a4-fbf2-42a3-9a87-319c9768
47 + <cell id="{c575d418-ea30-4f25-8cca-fbee2461
48 48 <labels>
```

- Side-by-side view should load.

Files Details

Parent 1 – This commit ▾ Filter (c2253a57-fbea-47bc-9032-b68a47655e01).xml -7 +7 /PermitManagement/Entities/contoso\_Permit/FormXml/main/(c2253a57-fbea-47bc-9032-b68a47655e01).xml Side-by-side ▾ ↑ ↓

|    |   |                                                                                   |      |    |                                                                                   |
|----|---|-----------------------------------------------------------------------------------|------|----|-----------------------------------------------------------------------------------|
| 42 |   | </labels>                                                                         | 42   |    | </labels>                                                                         |
| 43 |   | <control id="contoso_buildsite" classid="{270BD3DB-D9AF-4782-9032-b68a47655e01}"> | 43   |    | <control id="contoso_buildsite" classid="{270BD3DB-D9AF-4782-9032-b68a47655e01}"> |
| 44 |   | </row>                                                                            | 44   |    | </row>                                                                            |
| 45 |   | <row>                                                                             | 45   |    | <row>                                                                             |
| 46 |   | <cell id="{1ea0b6a4-fbf2-42a3-9a87-319c9768acae}" locklevel="0">                  | 46   |    | <cell id="{c575d418-ea30-4f25-8cca-fbee2461abc8}" locklevel="0">                  |
| 47 | - | <labels>                                                                          | 47 + |    | <labels>                                                                          |
| 48 | - | <label description="Contact" languagecode="1033" />                               | 48   | +> | <label description="Start Date" languagecode="1033" />                            |
| 49 | - | </labels>                                                                         | 49 + |    | </labels>                                                                         |
| 50 | - | <control id="contoso_contact" classid="{270BD3DB-D9AF-4782-9032-b68a47655e01}">   | 50   | +> | <control id="contoso_startdate" classid="{5B773807-9FB2-42D8-9032-b68a47655e01}"> |
| 51 | - | </row>                                                                            | 51 + |    | </row>                                                                            |
| 52 | - | <cell id="{c575d418-ea30-4f25-8cca-fbee2461abc8}" locklevel="0">                  | 52   |    | <cell id="{1ea0b6a4-fbf2-42a3-9a87-319c9768acae}" locklevel="0">                  |
| 53 | - | <labels>                                                                          | 53   |    | <labels>                                                                          |
| 54 | - | <label description="Start Date" languagecode="1033" />                            | 54   | +> | <label description="Contact" languagecode="1033" />                               |
| 55 | - | </labels>                                                                         | 55 + |    | </labels>                                                                         |
| 56 | - | <control id="contoso_startdate" classid="{5B773807-9FB2-42D8-9032-b68a47655e01}"> | 56   | +> | <control id="contoso_contact" classid="{270BD3DB-D9AF-4782-9032-b68a47655e01}">   |
| 57 | - | </row>                                                                            | 57 + |    | </row>                                                                            |
| 58 | - | <cell id="{1ea0b6a4-fbf2-42a3-9a87-319c9768acae}" locklevel="0">                  | 58   |    | <cell id="{c575d418-ea30-4f25-8cca-fbee2461abc8}" locklevel="0">                  |
| 59 | - | <labels>                                                                          | 59 + |    | <labels>                                                                          |
| 60 | - | <label description="Contact" languagecode="1033" />                               | 60   | +> | <label description="Start Date" languagecode="1033" />                            |
| 61 | - | </labels>                                                                         | 61   | +> | </labels>                                                                         |
|    |   | </cell>                                                                           |      |    | </cell>                                                                           |
|    |   | </row>                                                                            |      |    | </row>                                                                            |