HOL PCF Picklist

Hands-on Lab

Released:

Terms of Use

© 2019 Microsoft Corporation. All rights reserved.

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see **Microsoft Copyright Permissions** at [**http://www.microsoft.com/permission**](http://www.microsoft.com/permission)

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The Microsoft company name and Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

**This document reflects current views and assumptions as of the date of development and is subject to change.  Actual and future results and trends may differ materially from any forward-looking statements.  Microsoft assumes no responsibility for errors or omissions in the materials.**

**THIS DOCUMENT IS FOR INFORMATIONAL AND TRAINING PURPOSES ONLY AND IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT.**

“About the Authors” is an important item as it establishes the credibility of the text by explaining how much experience the author team has in the field. Add as many “About the Authors” as needed from the Quick Parts Gallery. Change “Author” to a different title, such as “Subject Matter Expert” if that is more appropriate. Include SMEs who assisted with the development of the course. Add this to every Module.

Table of Contents

[Lab Overview 1](#_Toc22046378)

[Learning Objectives 2](#_Toc22046379)

[Exercise 1: Create PCF Control 3](#_Toc22046380)

[Task 1: Install Microsoft PowerApps CLI and Prerequisites 3](#_Toc22046381)

[Task 2: Setup Components Project 3](#_Toc22046382)

[Task 3: Open the CLI in VS Code and Run 11](#_Toc22046383)

[Task 4: Deploy 13](#_Toc22046384)

[Task 5: Use the PCF Control 15](#_Toc22046385)

# Lab Overview

Several paragraphs – or longer – describing an overview of the lab including a description of the lab and why certain topics are covered. This is also called the ‘LAB ABSTRACT’ that will be used for hand-off to conferences as part of the content hand-off process

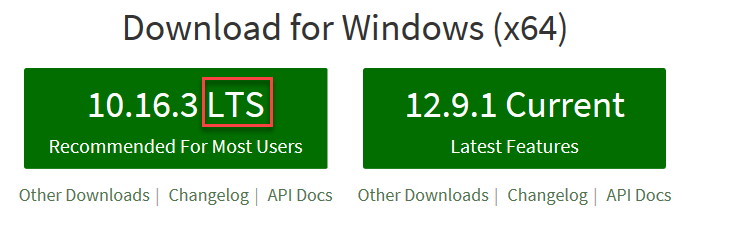
# Learning Objectives

* Understand page & navigation
* Use device capabilities
* Data sources and formulas to work with CDS
* Gallery and Forms
* Variables
* Binding of properties

# Exercise 1: Create PCF Control

### Task 1: Install Microsoft PowerApps CLI and Prerequisites

1. Navigate to <https://nodejs.org/en/> and select LTS version.
2. d make sure you have the correct environment selected.



1. Click **Run**.



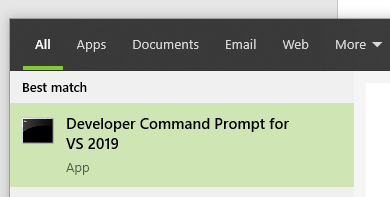
1. Follow the setup wizard steps to complete installing **Node.js**
2. Navigate to <https://dotnet.microsoft.com/download/dotnet-framework/net462> and select **Developer Pack**.



1. Click **Run**.
2. Follow the setup wizard steps to complete the installation.

### Task 2: Setup Components Project

1. Start **Developer Command Prompt**.



1. Run the command shown below to change directory. Replace **[Computer User Name]** with your OS user name

cd C:\Users\[Computer User Name]\Documents

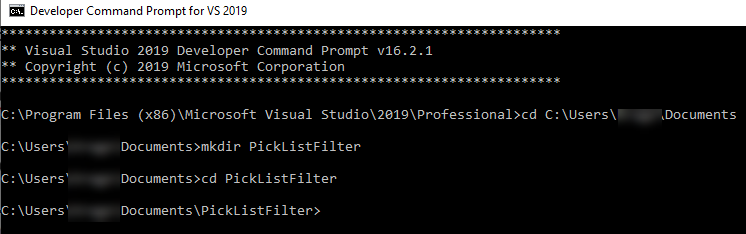
1. Create a new folder with the name PickListFilter.

mkdir PickListFilter

1. Change directory to the folder you just created.

cd PickListFilter

1. Your command prompt should now look like the image below.

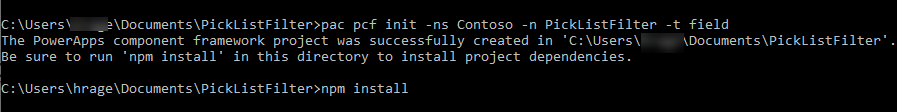


1. Initialize the component. This command will create a set of files that will implement a dataset component. You will customize these files as per your specific component as we continue.

pac pcf init -ns Contoso -n PickListFilter -t field

1. Install dependencies by running **npm install** command in the Terminal and wait for the dependencies to be installed.

npm install



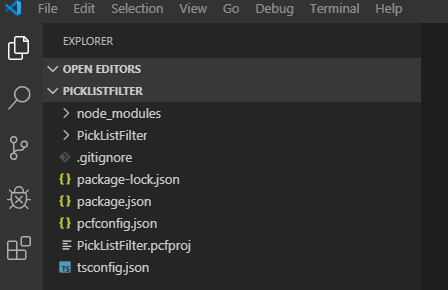
1. Add React office fabric to your project and wait for the installation to complete.

npm install react react-dom office-ui-fabric-react -save

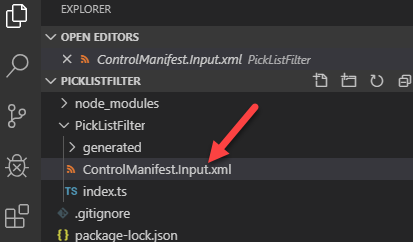
1. Open the project in **Visual Studio Code**.

code .

1. Your project structure should look like the image below.



1. Expand the **PiclListFilter** folder and click to open the **ControlManifest.Input.xml** file.



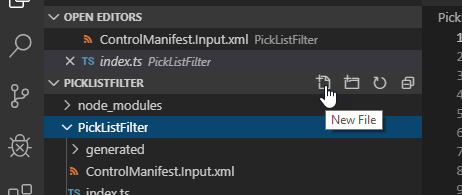
1. Locate the **Property** element and replace it with the xml fragment below.

<property name="picklistField" display-name-key="Picklist Field" description-key="Picklist Field to filter" of-type="OptionSet" usage="bound" required="true" />

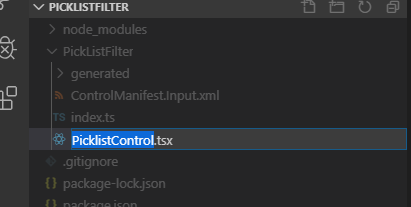
    <property name="filtering" display-name-key="Picklist Field" description-key="Picklist Field to filter" of-type="SingleLine.Text" usage="input" required="true" />



1. Select the **PickListFilter** folder and click **+ New File**.



1. Name the new file **PicklistControl.tsx**.



1. Open the **PickListControl** file you just created and add the below imports.

import \* as React from 'react';

import { Dropdown, IDropdownOption } from "office-ui-fabric-react/lib/Dropdown";

1. Add the below export.

export interface IControlProps {

    value: number | undefined,

    options: IDropdownOption[],

    onChange: (newValue: number | undefined) => void

}

1. Add the picklist control export.

export class PicklistControl extends React.Component<IControlProps> {

    constructor(props: IControlProps, context: any) {

        super(props);

    }

    private \_onChange = (event: React.FormEvent<HTMLDivElement>, item: IDropdownOption | undefined): void => {

        var selectedKey = item && item.key as number;

        this.props.onChange(selectedKey);

    };

    render() {

        return (

            <Dropdown

                selectedKey={this.props.value}

                onChange={this.\_onChange}

                placeholder="--Select--"

                options={this.props.options}

                styles={{ dropdown: { width: "100%" } }}

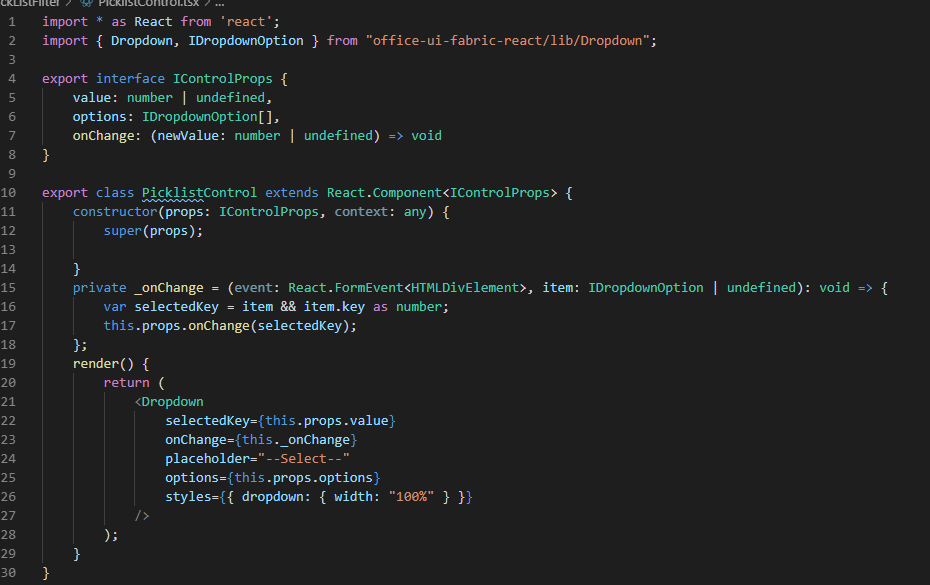
            />

        );

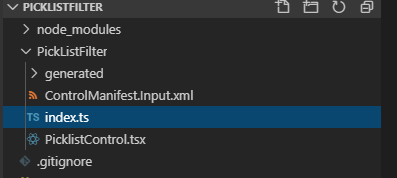
    }

}

1. Your **PicklistControl.tsx** file should now look like the image below.



1. Click to open the **index.ts** file.



1. Add the below **imports** to **index.ts** after the existing import.

import \* as React from "react";

import \* as ReactDOM from "react-dom";

import { PicklistControl } from "./PicklistControl";

import { IDropdownOption } from "office-ui-fabric-react/lib/Dropdown";

import { initializeIcons } from '@uifabric/icons';



1. Add the below constant above the class.

const picklistFieldName = "picklistField";

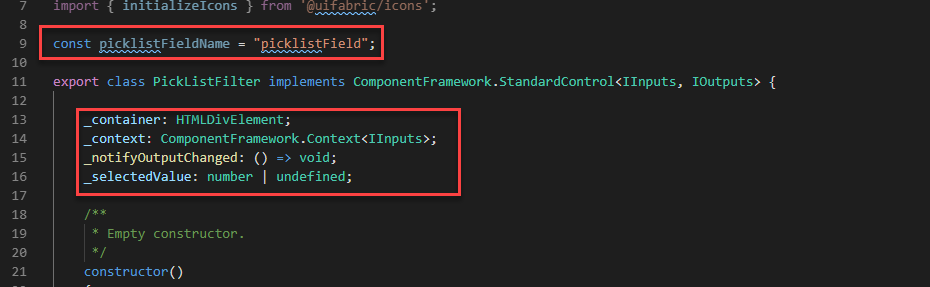
1. Add the below properties inside the class and before the constructor.

    \_container: HTMLDivElement;

    \_context: ComponentFramework.Context<IInputs>;

    \_notifyOutputChanged: () => void;

    \_selectedValue: number | undefined;



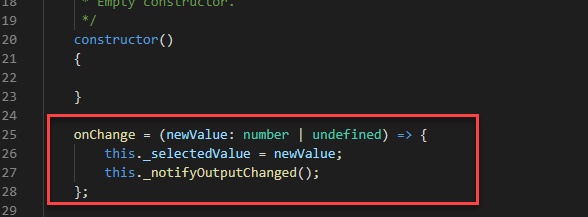
1. Add **onChange** function to the **index.ts** file.

    onChange = (newValue: number | undefined) => {

        this.\_selectedValue = newValue;

        this.\_notifyOutputChanged();

    };



1. Add the function that will render the control to the **index.ts** file.

    private renderControl(context: ComponentFramework.Context<IInputs>) {

        // Get options metadata

        var metadata = context.parameters.picklistField.attributes;

        if (metadata && context.parameters.filtering.raw) {

            // Get the configuration input parameter - comma separated list of optionset values

            var filterInput = context.parameters.filtering.raw.

                split(",").map((a) => { return parseInt(a) });

            // Create filtered options provided by the metadata

            // Do this with every render in case they change

            var options = metadata.Options

                .filter((o) => filterInput.includes(o.Value))

                .map((v) => {

                    return {

                        key: v.Value,

                        text: v.Label

                    } as IDropdownOption

                });

            ReactDOM.render(

                React.createElement(PicklistControl, {

                    value: this.\_selectedValue,

                    options: options,

                    onChange: this.onChange

                }),

                this.\_container

            );

        }

    }



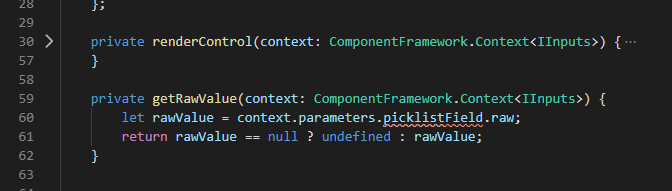
1. Add the function below to the index.ts file.

private getRawValue(context: ComponentFramework.Context<IInputs>) {

        let rawValue = context.parameters.picklistField.raw;

        return rawValue == null ? undefined : rawValue;

    }



1. Add the snippet below inside the **init** function.

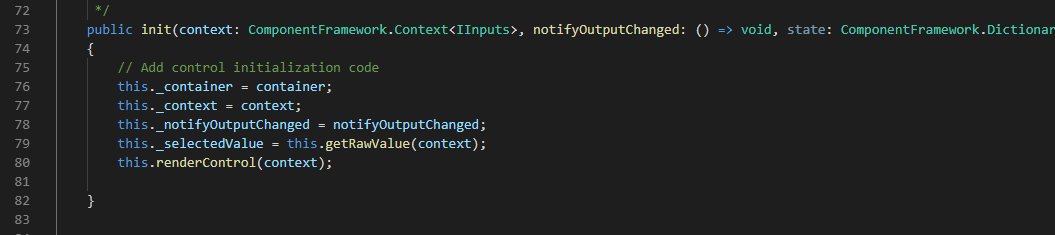
        this.\_container = container;

        this.\_context = context;

        this.\_notifyOutputChanged = notifyOutputChanged;

        this.\_selectedValue = this.getRawValue(context);

        this.renderControl(context);



1. Locate the **updateView** function and add the snippet below.

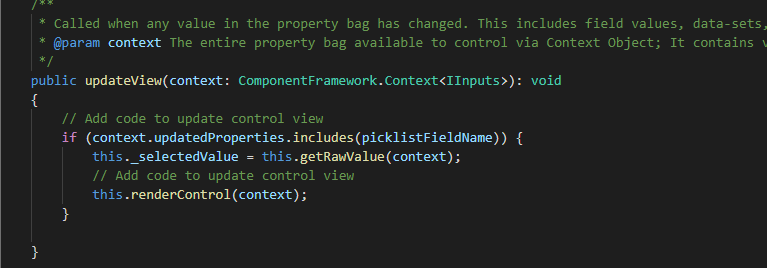
if (context.updatedProperties.includes(picklistFieldName)) {

            this.\_selectedValue = this.getRawValue(context);

            // Add code to update control view

            this.renderControl(context);

        }



1. Locate **getOutputs** and replace the **return {}** line with the snippet below.

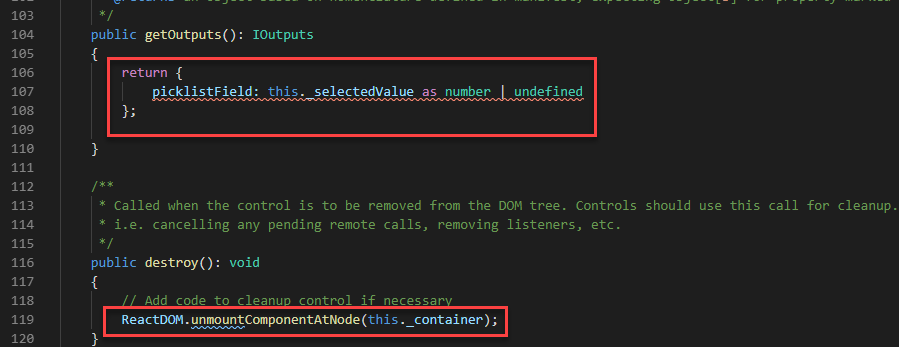
return {

            picklistField: this.\_selectedValue as number | undefined

        };

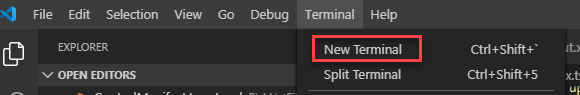
1. Locate the **destroy** function and add the snippet below.

ReactDOM.unmountComponentAtNode(this.\_container);

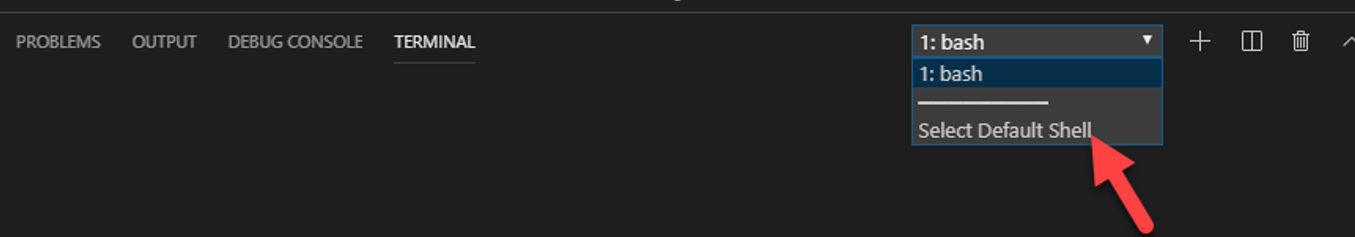


### Task 3: Open the CLI in VS Code and Run

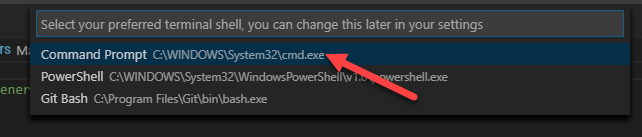
1. Click **Terminal** and select **New Terminal**.

****

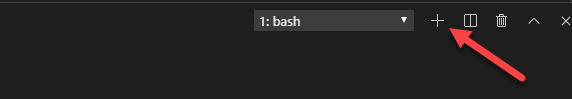
1. If **cmd** isn’t your **Default Shell**, click on the arrow and click **Select Default Shell**.



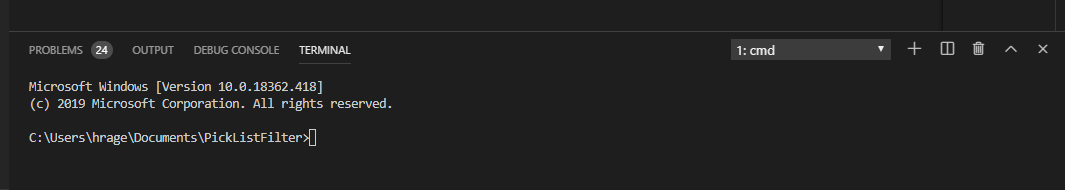
1. Select **Command Prompt**.



1. Click **New Terminal**.

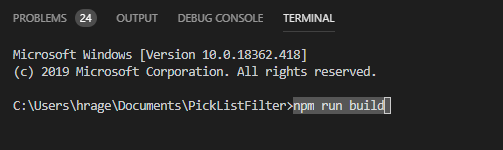


1. The **cmd** terminal should now open.

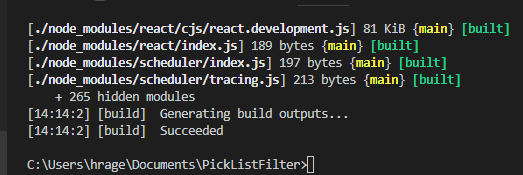


1. Run the **build** command.

npm run build



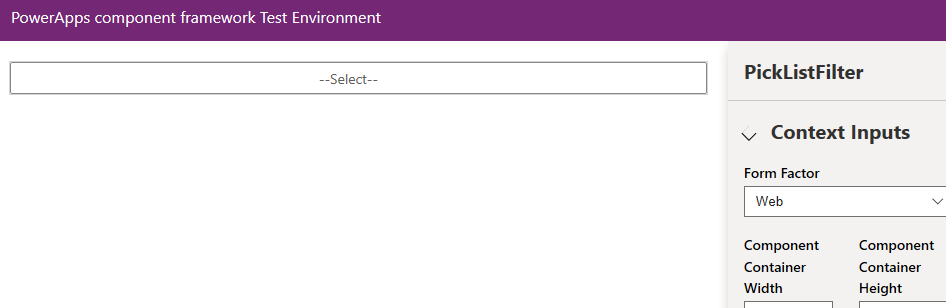
1. The build should succeed.



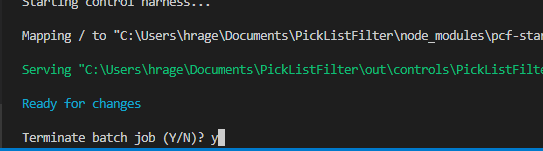
1. Run the **start** command.

npm start

1. The test environment should render the control, but you can’t interact with it.

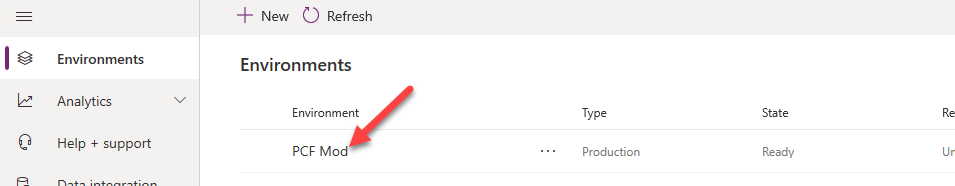


1. Close the Test Environment browser window or tab.
2. Go back to the terminal in Visual Studio code.
3. Click on the **Terminal** and press the **[CONTROL]** key and **c**
4. Type **y** and enter **[ENTER]**.

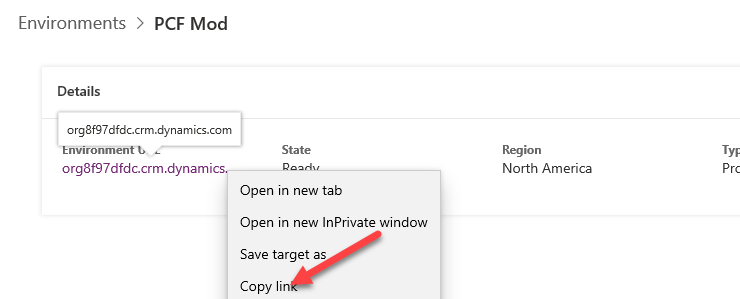


### Task 4: Deploy

1. Navigate to make <https://admin.powerplatform.microsoft.com/> select Environments and click to open the environment you are deploying the PCF control to.



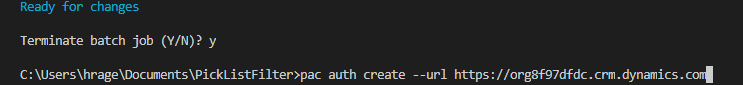
1. Right click on the **Environment URL** and select **Copy Link**.



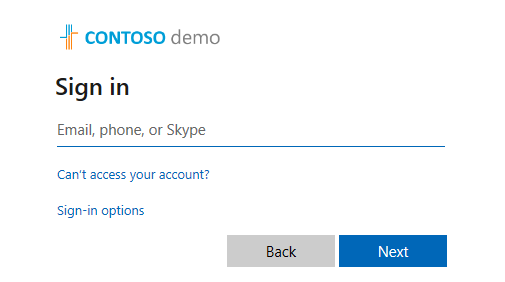
1. Go back to the Visual Studio terminal and type the command below.

pac auth create --url

1. Add the **URL** you copied in front of **url**.
2. Your command should now look like the image below. press [ENTER]

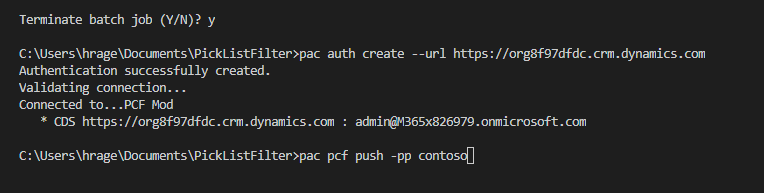


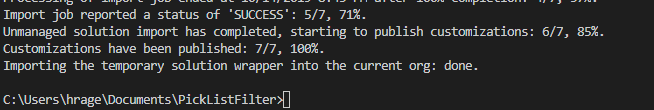
1. Login with your admin user credentials.



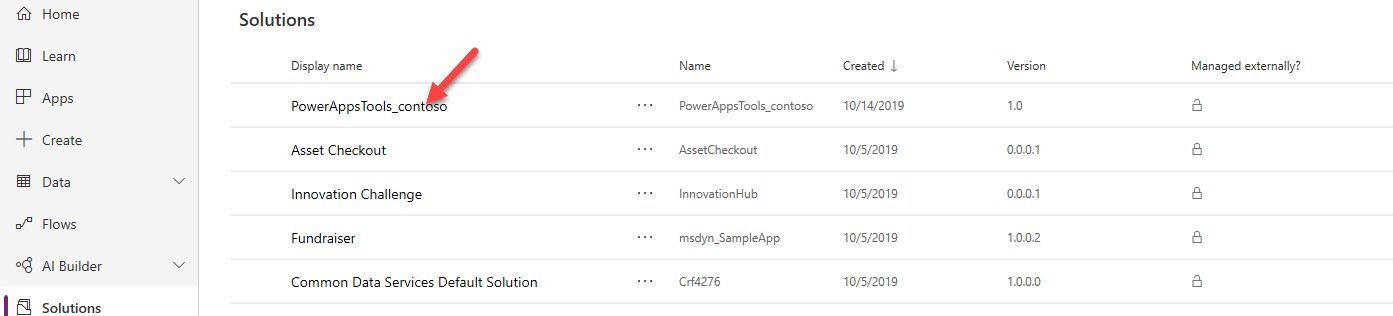
1. Run the command below and wait for the publishing to complete.

pac pcf push -pp contoso

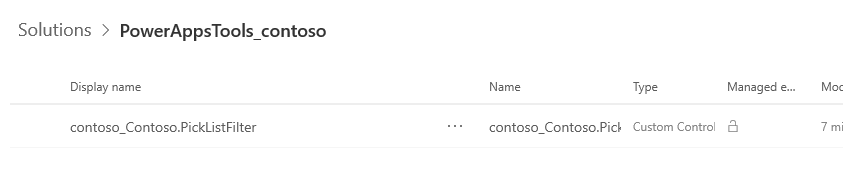




1. Navigate to <https://make.powerapps.com/> and select the correct environment.
2. Select **Solutions**.
3. You should see the solution **PowerApps** **Tools** created. Click to open the solution.

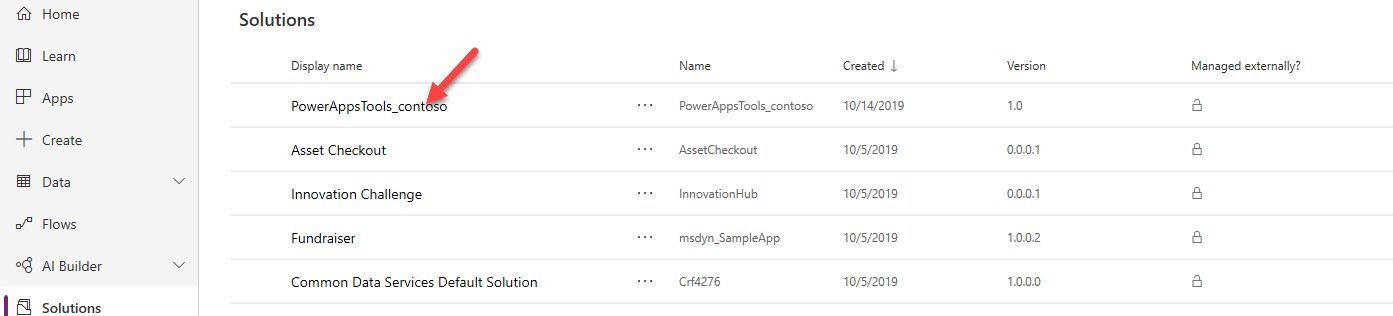


1. You should see the Custom Control you created.

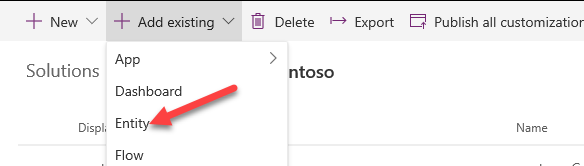


### Task 5: Use the PCF Control

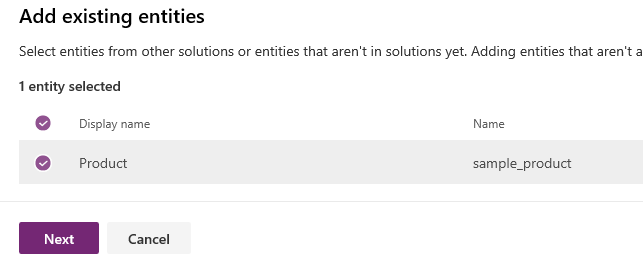
1. Navigate to <https://make.powerapps.com/> and select the correct environment.
2. Select **Solutions**.
3. You should see the solution **PowerApps** **Tools** created. Click to open the solution.



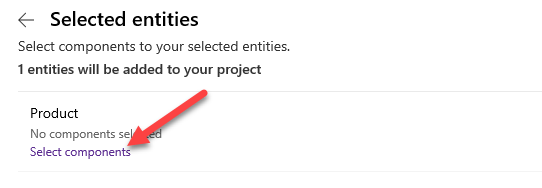
1. Click **+ Add Existing** and select **Entity**.



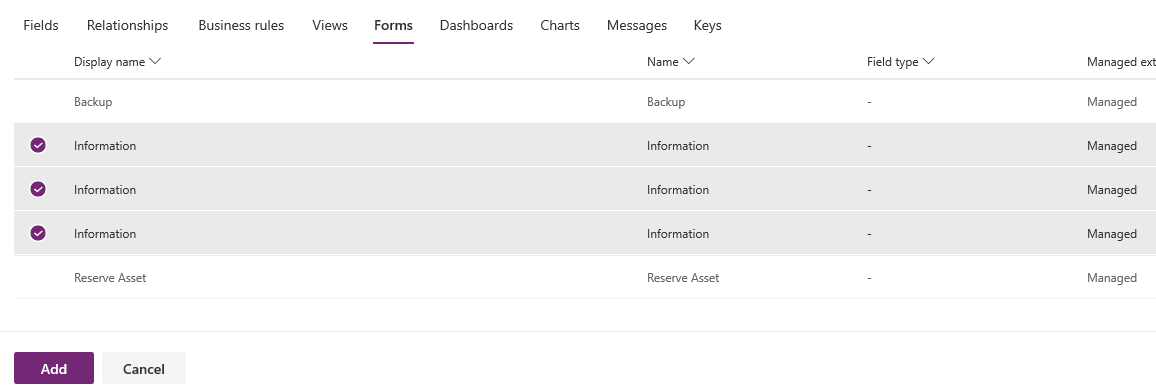
1. Select **Product** and click **Next**.



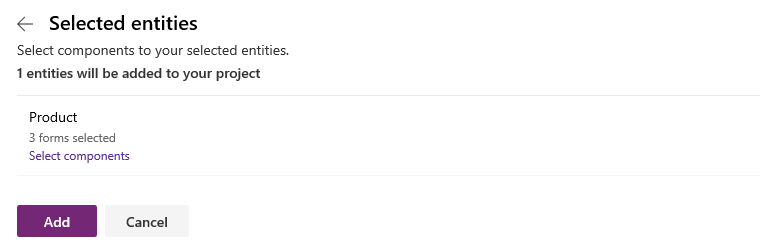
1. Click **Select Components**.



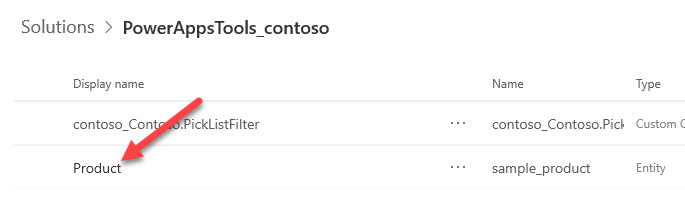
1. Select the **Forms** tab, select all the **Information** forms and click **Add**.



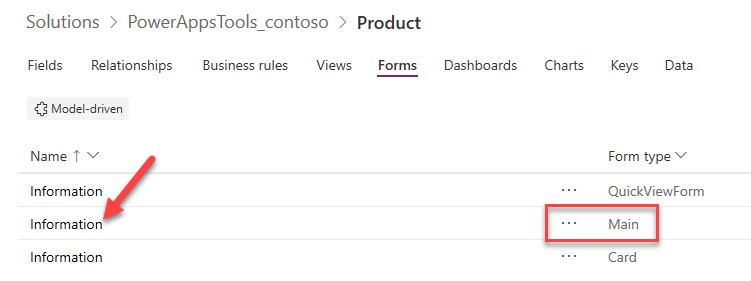
1. Click **Add** again.



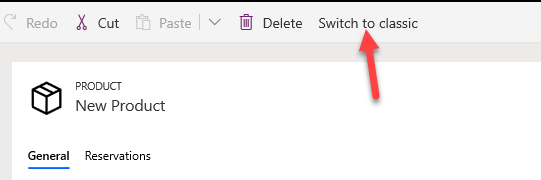
1. Click to open the **Product** entity you just added to your solution.



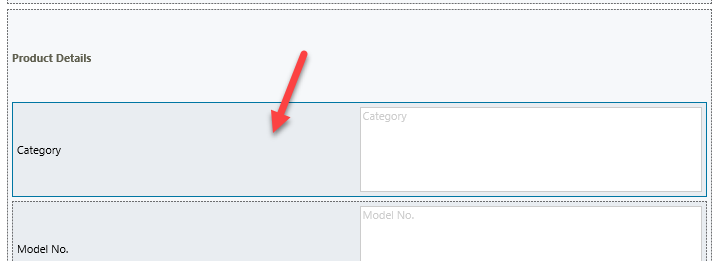
1. Select the **Forms** tab and click to open the **Main** form.



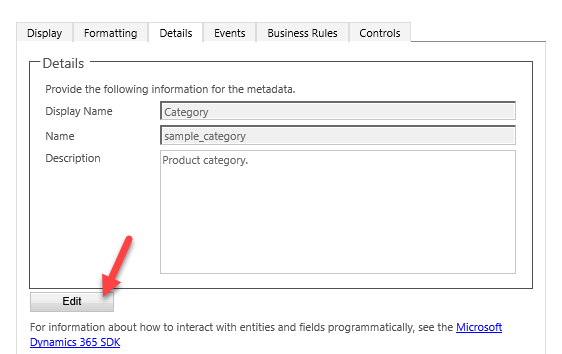
1. Click **Switch to Classic**.



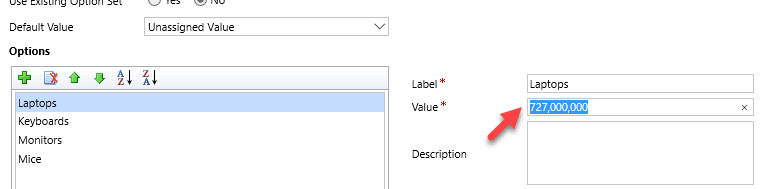
1. Double clikc the **Category** field.



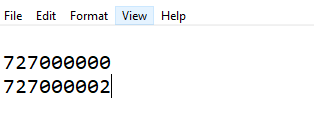
1. Select the **Details** tab and click **Edit**.



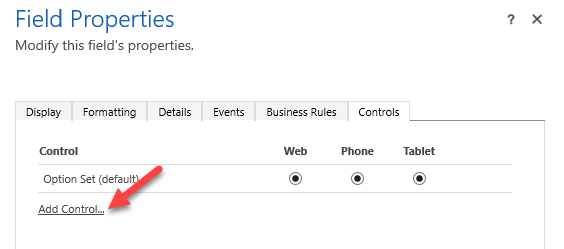
1. Scroll down to the **Options** and select **Laptops**.
2. Copy the **Value** and keep it in a notepad.



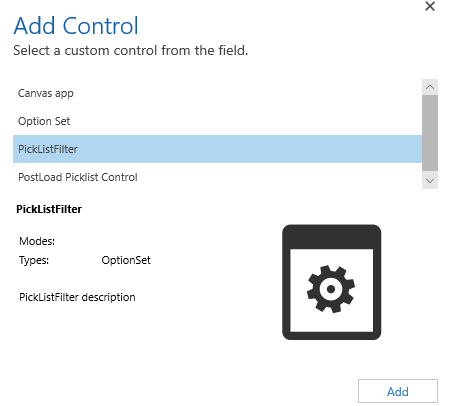
1. Select **Monitors** and copy the **Value** to a notepad.
2. You should now have two values in your notepad. Remove the commas.



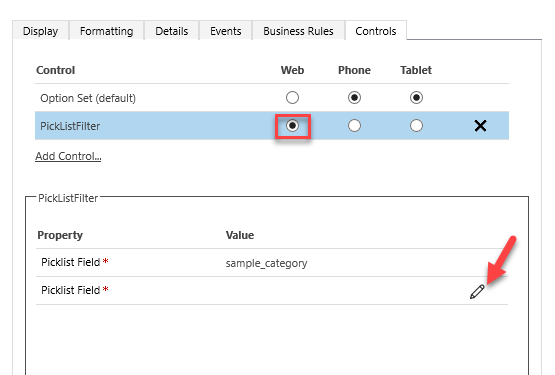
1. Go back to the field editor and close it.
2. Select the **Controls** tab and click **Add Control**.



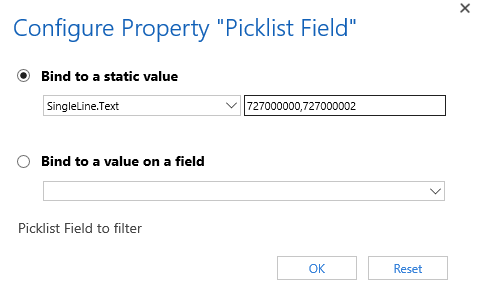
1. Select the **PickListFilter** you created and click **Add**.



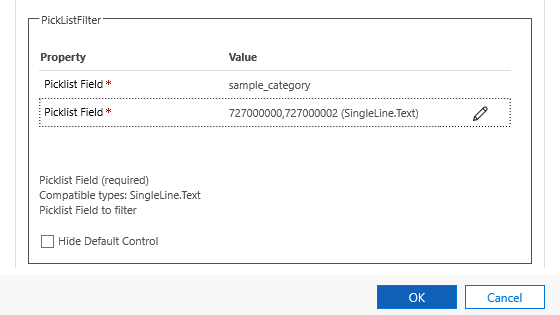
1. Select **Web** and click **Edit.**



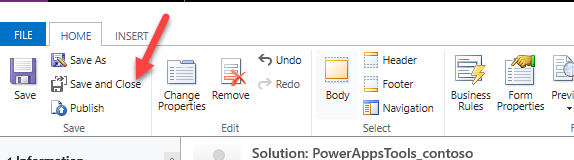
1. Select **Bind to Static Value**
2. Paste the two values you copied separated by a comma and click **OK**.



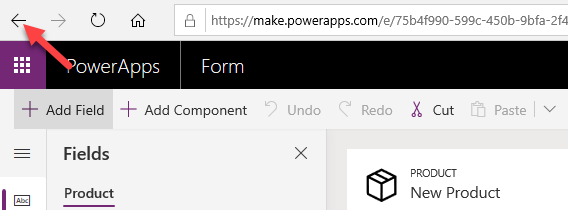
1. Click **OK** again.



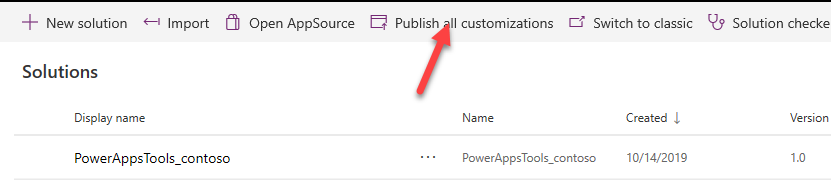
1. Click **Save and Close**.



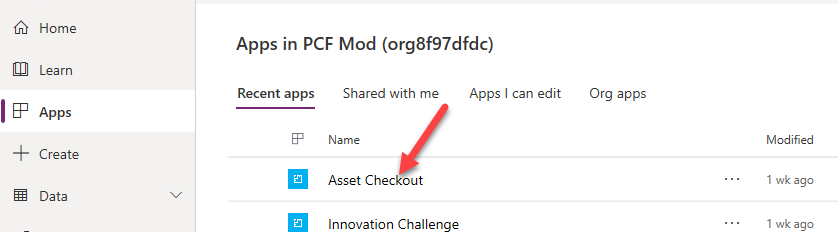
1. Click on the browser back button.



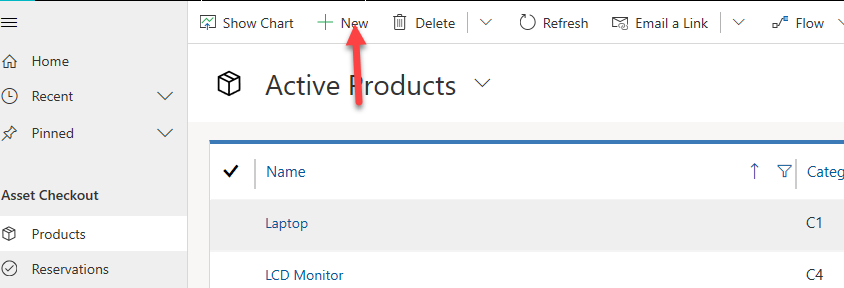
1. Select **Solutions** and click **Publish All Customizations**.



1. Select **Apps** and click to open the **Asset Checkout** application.



1. Select **Products** and click **+ New**.



1. The **Category** option-set should now have only the two options you provided.

