INTRODUCTION TO MICROSOFT POWERAPPS FOR ACCESS WEB APPS DEVELOPERS

ABSTRACT

The one constant in software technology is that it's constantly changing. You might ask: "What alternative is there to create a mobile or online solution for Microsoft Access?" Microsoft believes that Microsoft PowerApps is the answer. Although PowerApps is a relatively new product, Microsoft is making a significant investment in PowerApps to make it a premiere tool for business solutions and is adding new features on a regular basis.

The purpose of this white paper is to demonstrate that PowerApps is a successor tool worthy of consideration for an Access web apps power user and developer. What follows is an in-depth comparison of PowerApps and Access Web apps that examines the pros, cons, and similarities of each tool, their key features, and the way data is managed. Hopefully, you'll take the time to try out PowerApps and add it to your business solutions toolkit.

Ben Clothier, Andy Tabisz

March, 2017

What is Microsoft PowerApps?

In a nutshell, PowerApps is a software platform for creating and sharing corporate, line-of-business apps quickly and without writing code. It's designed to work natively on iPhone, Windows 10 Mobile, and Android phones, and works on the web as well. PowerApps allows you to connect to multiple data sources at once, allowing you to interact with this combined information as needed. This white paper assumes that you have basic knowledge of PowerApps. For more information, see Microsoft PowerApps Guided Learning.

PowerApps is designed for a corporate environment in which you share apps with employees. You sign in by using your work or school account. You can't use personal email addresses with domains, such as hotmail.com, gmail.com, or aol.com, because those are not associated with a corporate environment. For details, see Azure Active Directory authentication, which is used by Office 365.

What makes PowerApps different from Access web apps?

PowerApps and Access web apps are both geared towards mobility, but each platform has its own strengths and weaknesses. PowerApps can do many of the features of Access web apps, but it can't be viewed as a direct replacement for Access web apps.

Mobile-first

By far, the best feature of PowerApps is that it is designed with a mobile-first strategy, meaning that it is ideal for phone devices. Access web apps is a browser-based application which does not work well on smart phones.

Multiple data sources

PowerApps can connect to many data sources including SharePoint, Office 365, Dropbox, Salesforce, MailChimp, and dozens of other data connections. Just like Power BI, each app you create can utilize data from multiple sources. By contrast, Access web apps are housed completely inside of SharePoint, whether in the cloud or on-premises, although you can interact with SharePoint data from an Access web app, and that data can be shared across Access web apps. You cannot connect one Access web app data to another Access web app. This document discusses how to replace an Access web app with a PowerApps app using SharePoint lists as the data source.

No-Code interface

Both are no-code solutions and neither uses VBA. Designing PowerApps screens is a lot like the Access web apps design experience except that instead of macros, you use extensive formulas and control properties which are much more like Microsoft Excel. This requires a different approach to writing logic compared to macros in Access web apps, which is addressed later. For more information on formulas, see Getting started with formulas and Formula reference for PowerApps.

Client-side vs server-side functionality

PowerApps functionality is mostly executed on the client-side. PowerApps may support some level of delegation where a bulk update is applied directly on the server. However, at the time of this writing, delegation is not supported for SharePoint lists. This means that when you work with more than one record, be careful not to write logic that could update only a subset of the list. Remember that PowerApps has a limit of 500 records downloaded to the local cache, and that's even before a filter is applied. The other consequence to consider is that if you edit a large amount of records, it requires more time for PowerApps to apply the changes back to the database; PowerApps currently can't invoke a procedure directly in the database. For more information on delegation, see Understand delegation.

By contrast, Access web apps does all query processing on the server, using a very powerful SQL server query engine. Access web apps have client-side, user interface (UI) macros, and server-side data macros that you can call. These data macros are essentially SQL triggers that perform backend operations, such as looping through all records in a dataset and performing a bulk update.

Although not covered in this white paper, PowerApps can easily work with Microsoft Flow, which allows you to create automation triggers, such as emailing to you when a single record is added but only if you were not the user that added the record.

Other important differences

Other differences are worth noting and factor into your decision-making as a developer.

Development environment

Access web apps require you to own Microsoft Access to do the development. With PowerApps, you have several development environments: PowerApps Mobile, which is available from the Apple App store, Google Play, and Windows store, the PowerApps Studio download (Windows 8.1 or higher), or the web interface. These are free and available at https://powerapps.microsoft.com. To author in PowerApps Studio requires a Windows 8.1 or higher device.

Understanding the distribution of native iOS and Android applications

To allow users to install iPhone/iPad and Android apps in most development software, each app must be submitted for approval to the app store. This is not necessary with PowerApps, because the PowerApps framework is already approved, and you're just running different apps under it. Think of it like this: when you use Excel, Excel doesn't care what kind of spreadsheet you are viewing, whether it's a budget spreadsheet, a customer service log, or employee overtime form. In the same manner, the PowerApps app is already certified and you can run any of the apps you create without going through an app store approval process.

The Common Data Service

When reading about PowerApps, you'll undoubtedly see references to the Microsoft Common Data Service (CDS). CDS is the Microsoft Azure–based business application model and storage mechanism for the Microsoft business application platform. Together with gateways and connectors, it forms the basis of business solutions that are created by using Microsoft Power BI, Microsoft PowerApps, and Microsoft Flow. CDS provides common business entities and integration capabilities for importing data from multiple sources, with the goal of bridging the data gap between software as a service (SaaS) workloads and business suites. By using CDS, you can create analytics that span these separate workloads and suites. CDS provides a rich and productive development platform.

The CDS business entities are analogous to the nouns presented as starting point tables in Access web apps. And just like the nouns in Access web apps, These CDS business entities have built-in interrelationships that you can add and "hook up" at any time. These entities can be extended, and completely custom new entities can be created. PowerApps can automatically create simple multi-screen applications from data entities stored in the Common Data Service. For more information, see Common Data Service and Common Data Model overview.

Security

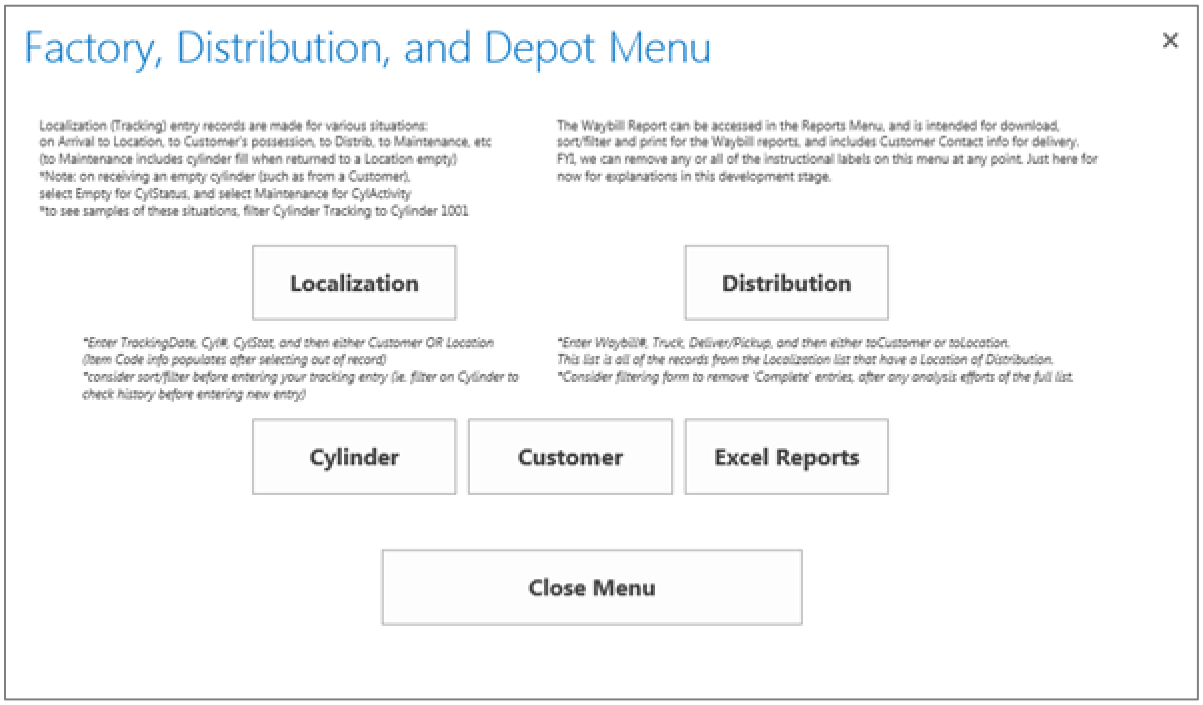
With Access web apps, you provide security to your app through SharePoint. You can also invite users outside your domain to use the app. With PowerApps, security resides with the data source you are connected to, whether you are using Dropbox, SharePoint Lists, or SQL Azure, and the requirement to sign in with your work or school account. Our examples use SharePoint Lists to connect to a data source. So, any user that you share your PowerApps with must sign in with a work or school account with sufficient permissions to the lists used by the PowerApps. Note that the precise security requirement for other data sources might be different. Consult the documentation on data sources for PowerApps if you plan to use different data sources.

Replicating the functionality of Access web apps in PowerApps

As mentioned previously, PowerApps is not a direct replacement for Access web apps. But there is a lot of functionality that you can either replicate or replace.

Menus and navigation

With Access web apps, there is built-in Navigation, but you can also use traditional menu choices. Here is an example based on a factory application for cylinders.



Factory, Distribution, and Depot Menu

Localization (Tracking) entry records are made for various situations:
on Arrival to Location, to Customer's possession, to Distrib, to Maintenance, etc
(to Maintenance includes cylinder fill when returned to a Location empty)
*Note: on receiving an empty cylinder (such as from a Customer),
select Empty for CylStatus, and select Maintenance for CylActivity
*to see samples of these situations, filter Cylinder Tracking to Cylinder 1001

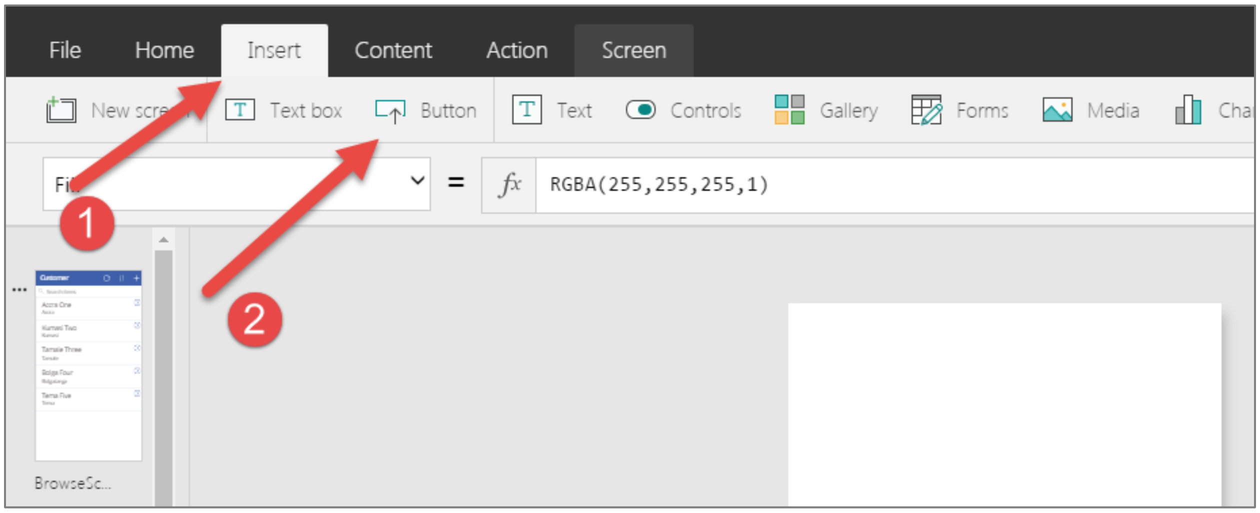
The Waybill Report can be accessed in the Reports Menu, and is intended for download, sort/filter and print for the Waybill reports, and includes Customer Contact info for delivery.
FYI, we can remove any or all of the instructional labels on this menu at any point. Just here for now for explanations in this development stage.

**Enter TrackingDate, Cyl#, CylStat, and then either Customer OR Location*
(Item Code info populates after selecting out of record)
**consider sort/filter before entering your tracking entry (ie. filter on Cylinder to check history before entering new entry)*

**Enter Waybill#, Truck, Deliver/Pickup, and then either toCustomer or toLocation.*
This list is all of the records from the Localization list that have a Location of Distribution.
**Consider filtering form to remove 'Complete' entries, after any analysis efforts of the full list.*

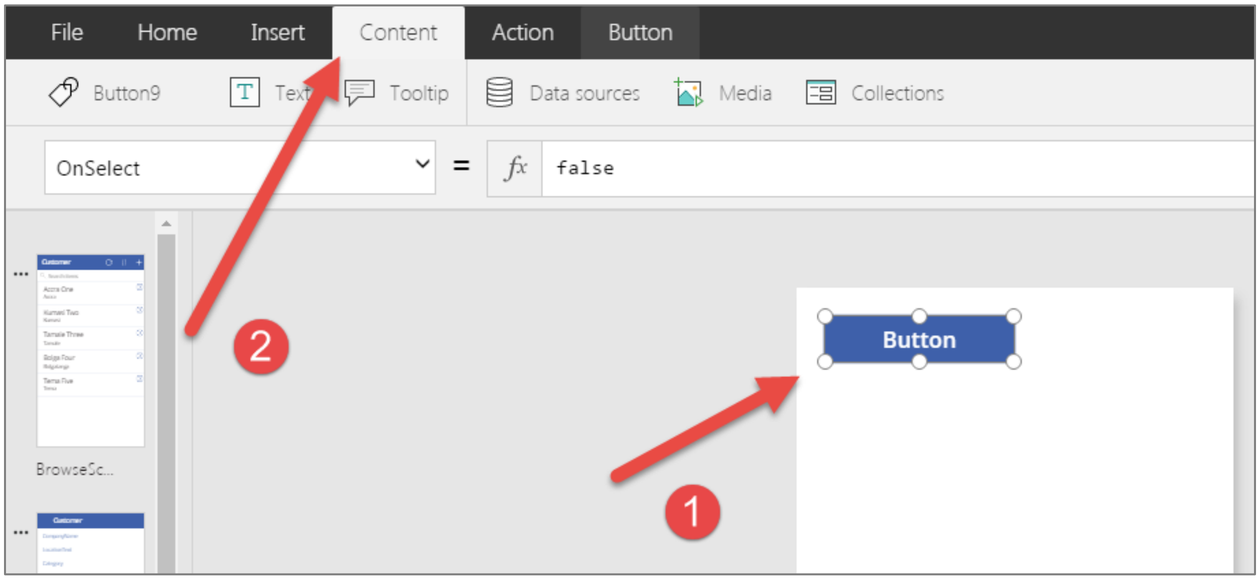
In this case, you can easily replicate the menu using a PowerApps screen. The biggest downside to a menu is planning for the mobile phone environment. You wouldn't use all the text, but you can certainly add a button which describes each section.

Adding several buttons in PowerApps is a little different. Whenever you add a button, PowerApps anticipates that you will immediately change the function for it as the following two illustrations show:



```
RGBA(255,255,255,1)
```

1. Select the **Insert** tab.
2. Select **Button**.

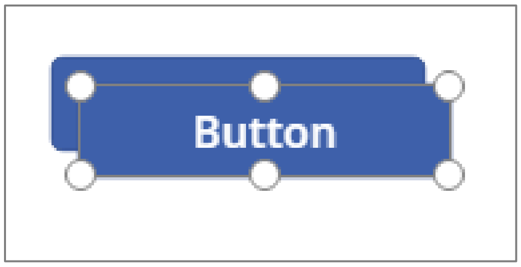


```
false
```

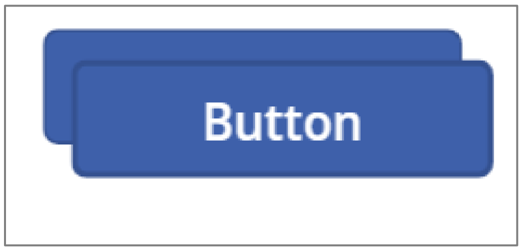
1. PowerApps adds the button.
2. PowerApps automatically changes the tab to **Content**.

Secondly, if you create multiple buttons, they overlap. In mobile apps, this should never happen. And Access web apps doesn't allow it to happen.

Here are the buttons in design mode:



Here are the buttons in preview mode:



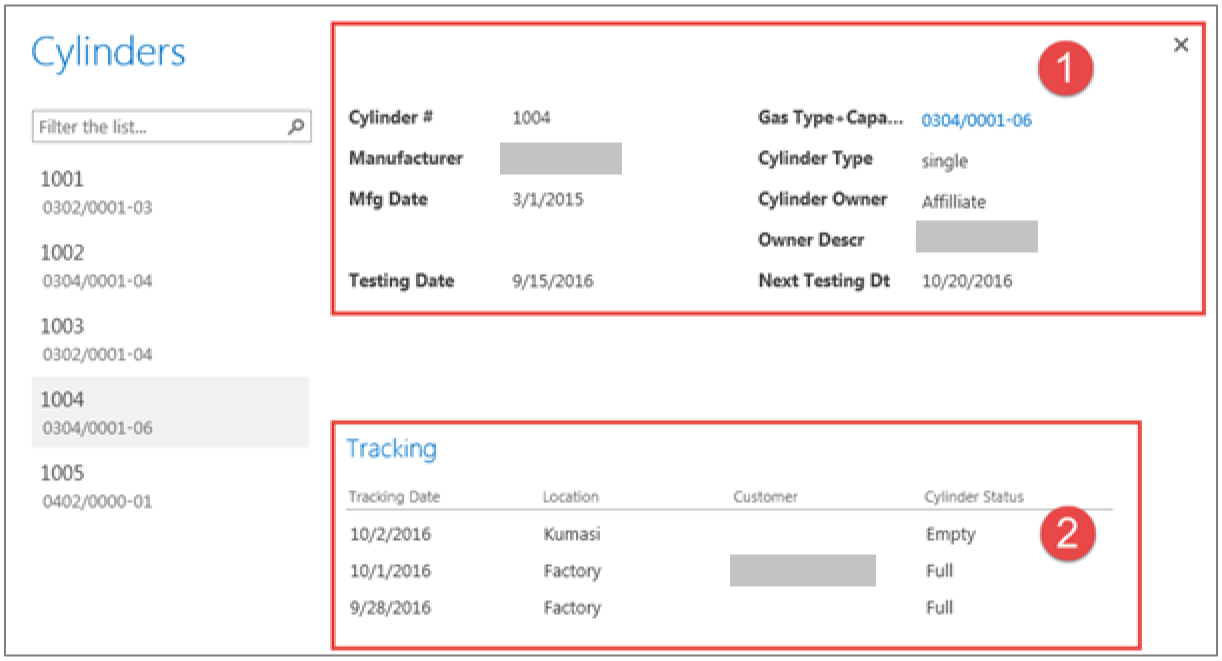
Other than that, it's easy to add the buttons, and resize and align them any way you need. This is like PowerPoint.

Form and layout

Access web apps have Blank views, List views and Datasheet views. Both Access web apps and PowerApps have blank views.

List views

With Access web apps, The List view includes a search bar as well as the record selected and often has a related items control. When you edit a record, you are not taken to a different view. It just becomes editable.



Cylinders

Filter the list...

1001
0302/0001-03

1002
0304/0001-04

1003
0302/0001-04

1004
0304/0001-06

1005
0402/0000-01

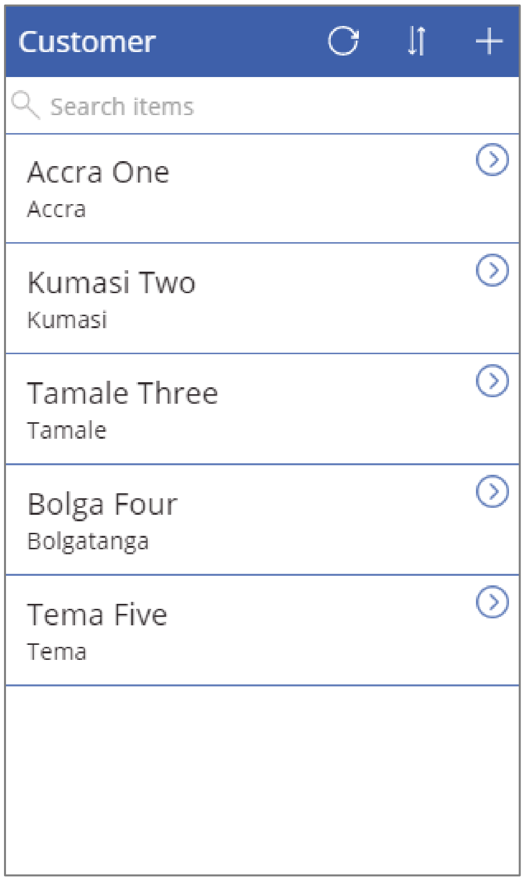
Cylinder #	1004	**Gas Type+Capa...**	0304/0001-06
Manufacturer	[redacted]	**Cylinder Type**	single
Mfg Date	3/1/2015	**Cylinder Owner**	Affilliate
		Owner Descr	[redacted]
Testing Date	9/15/2016	**Next Testing Dt**	10/20/2016

Tracking

Tracking Date	Location	Customer	Cylinder Status
10/2/2016	Kumasi		Empty
10/1/2016	Factory	[redacted]	Full
9/28/2016	Factory		Full

1. An Access web app main form.
2. An Access web app related items control.

With PowerApps, your List view is separated from the single record and it is replaced with a browse screen for which you have many layout options that are not included in Access Web apps.



Customer

Search items

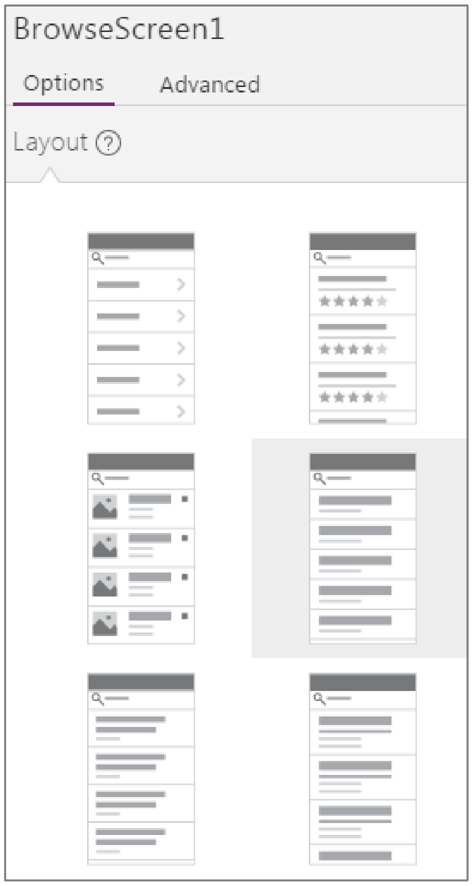
Accra One
Accra

Kumasi Two
Kumasi

Tamale Three
Tamale

Bolga Four
Bolgatanga

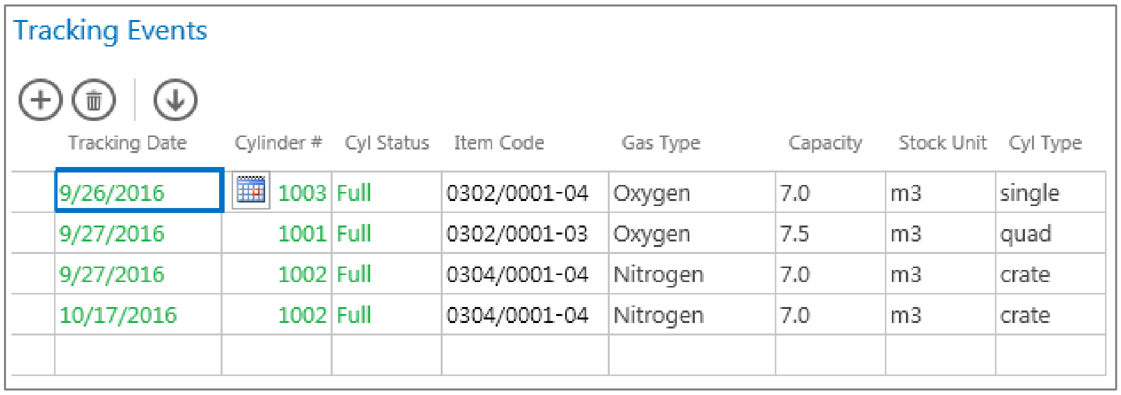
Tema Five
Tema



To replicate an Access web app form with a related item control in PowerApps, use the App from Data feature which provides basic integration between galleries and forms. For more information, see Generate an app to manage data in a SharePoint list. Then, you can customize the results. For more information, see Customize a layout in PowerApps and Customize forms in PowerApps.

Datasheet views

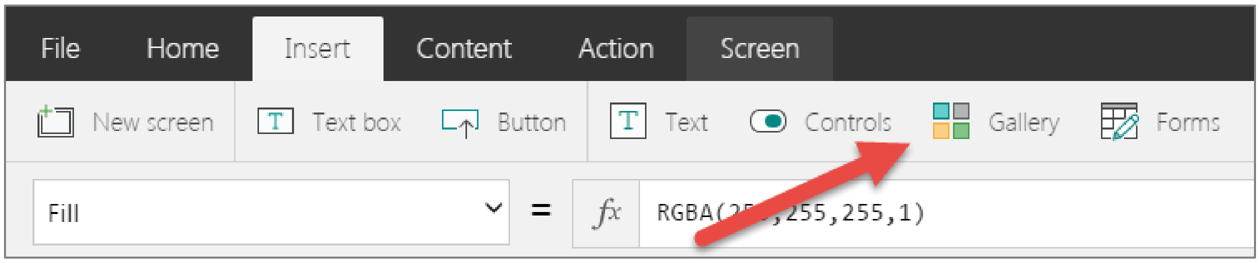
Access web apps have datasheet views, whereas PowerApps do not. But, there is a way that you can create a similar user experience. The following is a sample datasheet in Access web apps:



Tracking Events

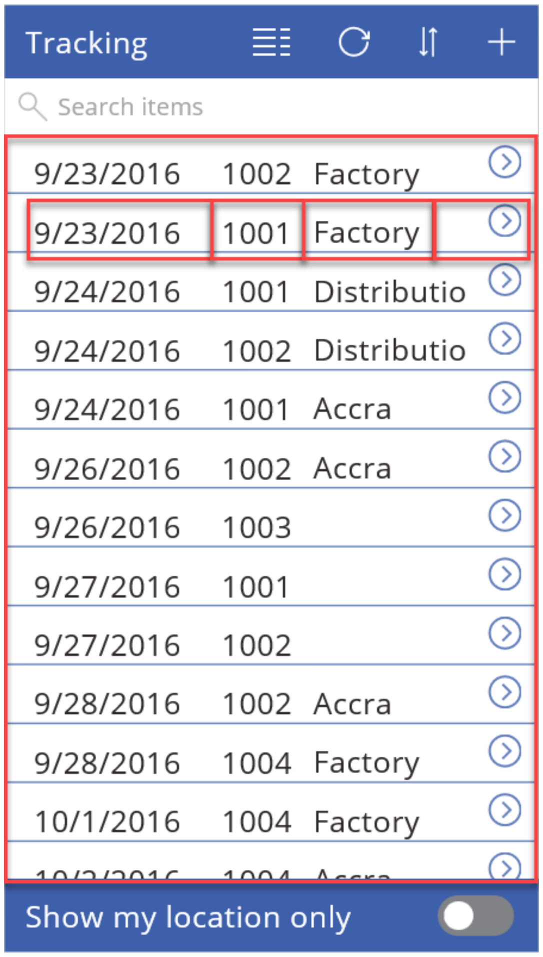
Tracking Date	Cylinder #	Cyl Status	Item Code	Gas Type	Capacity	Stock Unit	Cyl Type
9/26/2016	1003	Full	0302/0001-04	Oxygen	7.0	m3	single
9/27/2016	1001	Full	0302/0001-03	Oxygen	7.5	m3	quad
9/27/2016	1002	Full	0304/0001-04	Nitrogen	7.0	m3	crate
10/17/2016	1002	Full	0304/0001-04	Nitrogen	7.0	m3	crate

In PowerApps, there is a feature called Galleries which can be used in place of Access web apps datasheets:



RGBA(255,255,255,1)

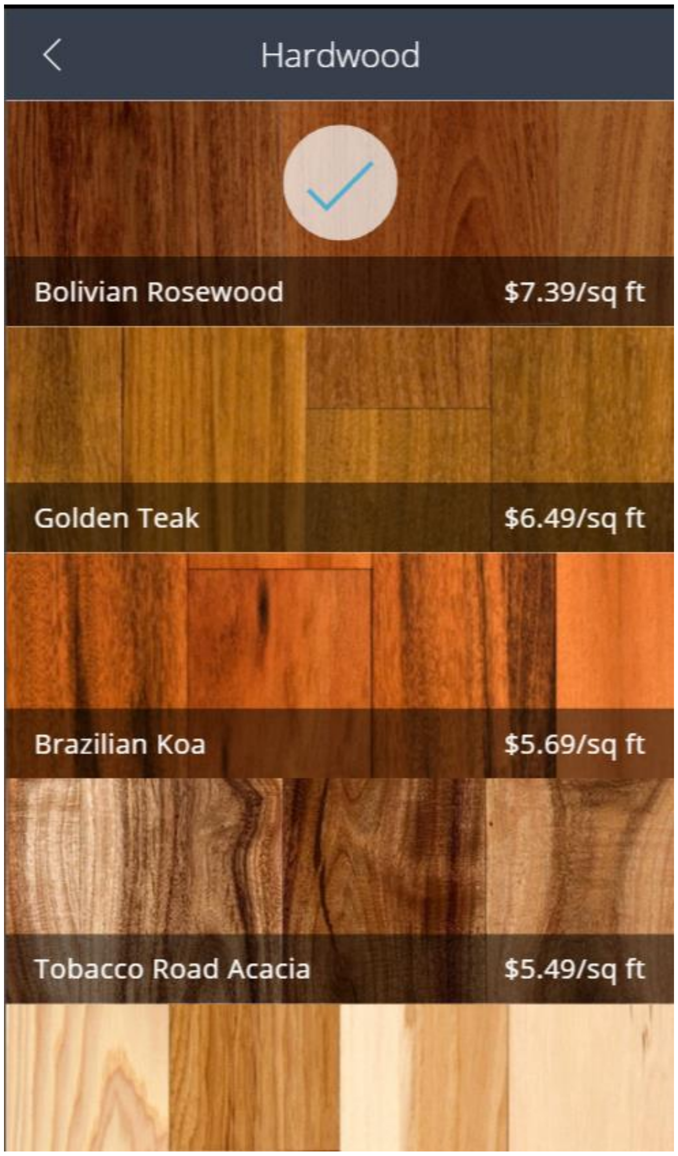
You can also place controls horizontally on a vertical gallery, as shown here:



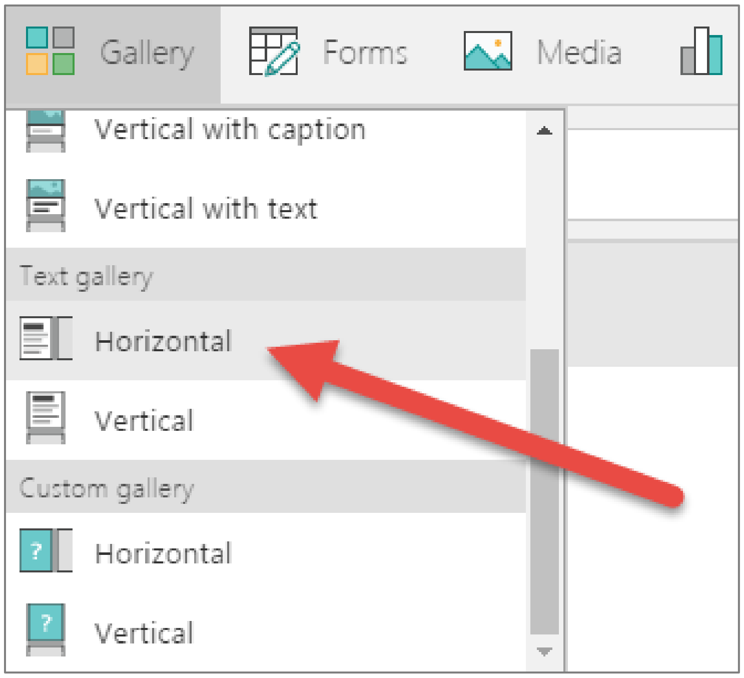
9/23/2016	1002	Factory
9/23/2016	1001	Factory
9/24/2016	1001	Distributio
9/24/2016	1002	Distributio
9/24/2016	1001	Accra
9/26/2016	1002	Accra
9/26/2016	1003	
9/27/2016	1001	
9/27/2016	1002	
9/28/2016	1002	Accra
9/28/2016	1004	Factory
10/1/2016	1004	Factory

This provides you with something like the layout of datasheet. However, unlike Access web apps datasheets, you cannot extend beyond the width of the screen. Also, you cannot use a horizontal layout because this would make the rows turn into columns. If you need additional space, you should use additional screens to provide the necessary real estate for extra data.

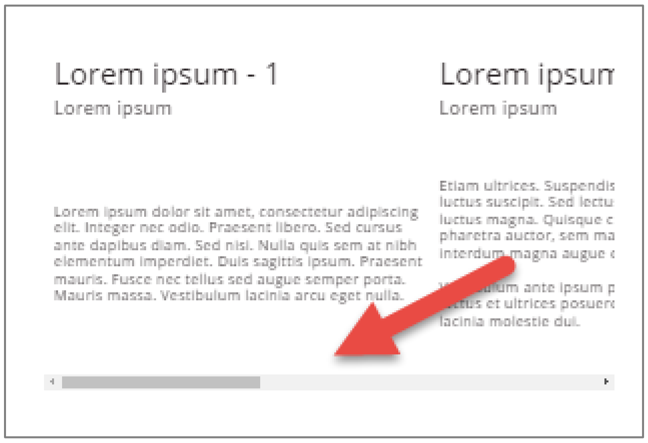
A gallery is often used for text-based lists, but also to show your pictures in a strip, so you can easily swipe through them. Here's an example of a cost estimating app where the salesperson selects from samples of flooring.



There is a choice under Gallery that allows you to select just a Text Gallery, either vertically or horizontally:



The following example shows only text and includes a scrollbar for you to slide through the records:


Lorem ipsum - 1
Lorem ipsum
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.
Lorem ipsum
Lorem ipsum

Why are there forms and galleries? Forms are intended to fit within a screen, whereas galleries have scrollbars that allow you to scroll through more records, either vertically or horizontally. When you create browse screens, you can use galleries to make it easy to load more records than can be shown in a single screen, and forms to edit a single record.

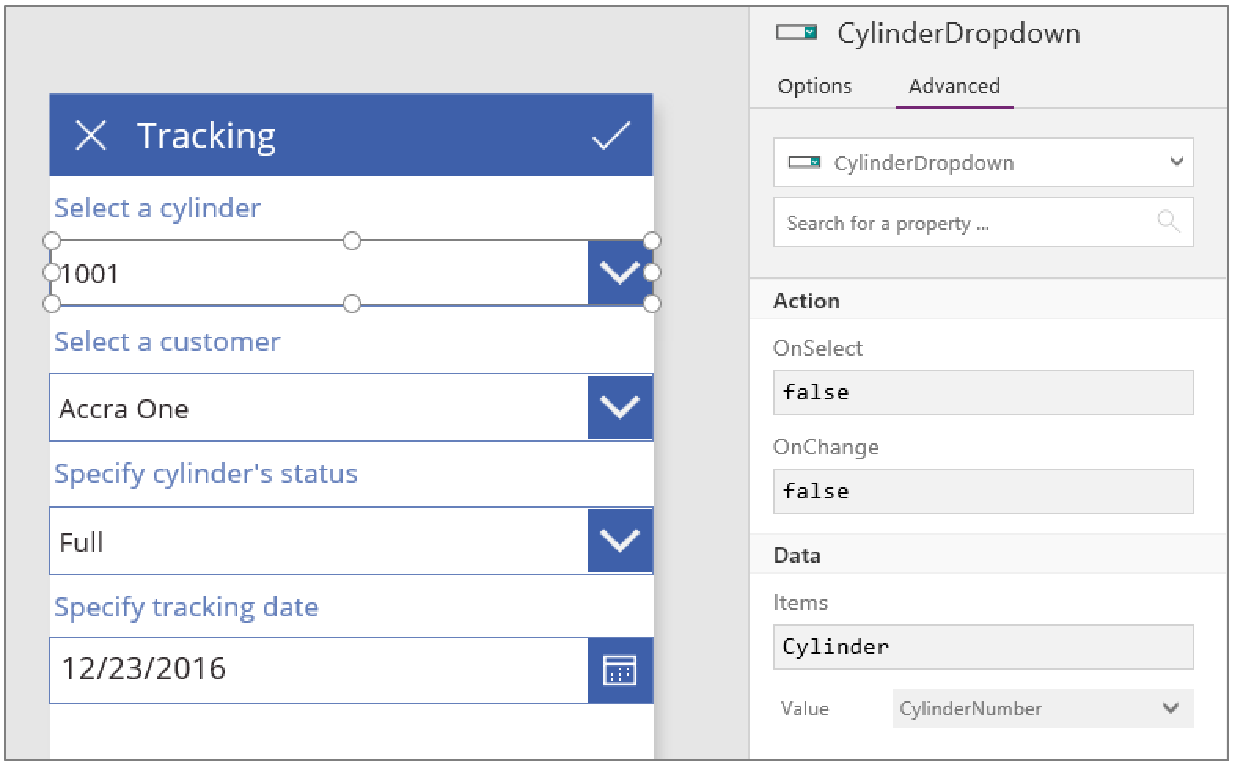
Related data

When you create a PowerApps app using the wizard, it is linked to only one data source. By contrast, Access web apps could have more than one table and often have relationships between tables. When you migrate the data from an Access web app, you most likely want to preserve the relationships. However, at the time this white paper was published, PowerApps makes no explicit assumptions about the relationships. Here are two different approaches for enabling related data.

Approach one: using lookups

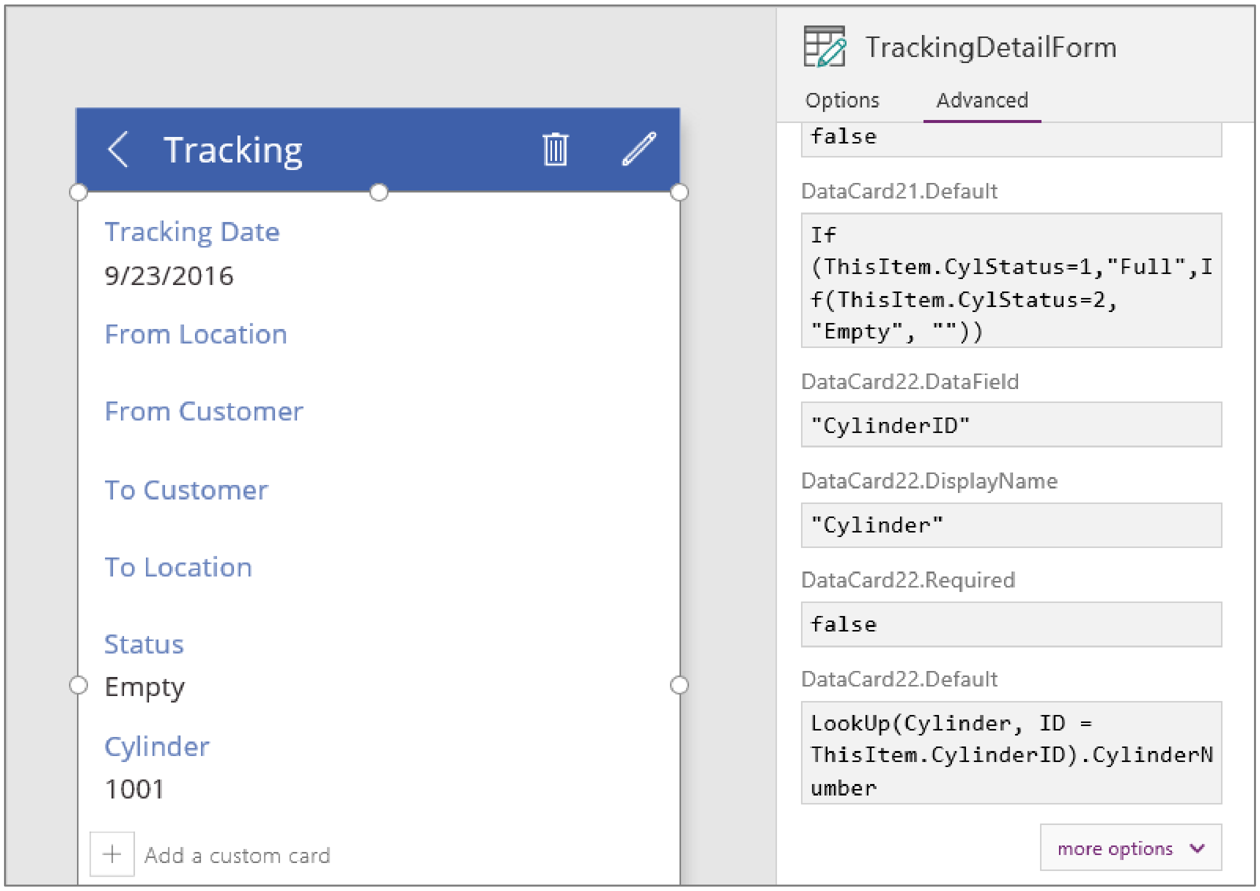
Suppose that on a form for tracking cylinders, you want to enable users to select from a list of cylinders. The list contains CylinderID, but the user expects to see Cylinder Number instead. In this case, you could use a dropdown control.

Add the dropdown control, set the Data source property to Cylinder, and then set the Display value property to CylinderNumber as the following example shows:



As you can see, this is similar to how it works in Access web apps and so is straightforward. However, when you create a display-only form and want to display the lookup value instead of CylinderID, you need to use a formula. In this case, on a display form, add a textbox and for the Default property of the data card, use the following formula:

LookUp(Cylinder, ID = ThisItem.CylinderID).CylinderNumber



```
false
```

DataCard21.Default

```
If
(ThisItem.CylStatus=1,"Full",I
f(ThisItem.CylStatus=2,
"Empty", ""))
```

DataCard22.DataField

```
"CylinderID"
```

DataCard22.DisplayName

```
"Cylinder"
```

DataCard22.Required

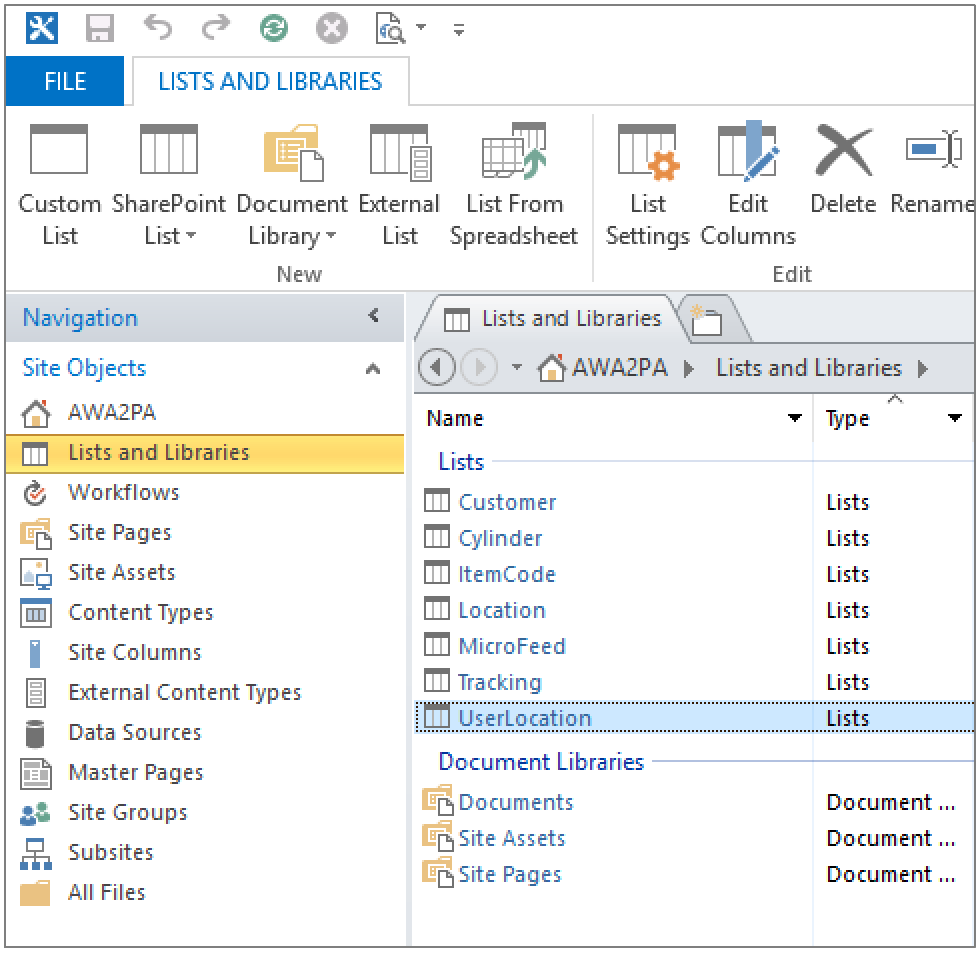
```
false
```

DataCard22.Default

```
LookUp(Cylinder, ID =
ThisItem.CylinderID).CylinderN
umber
```

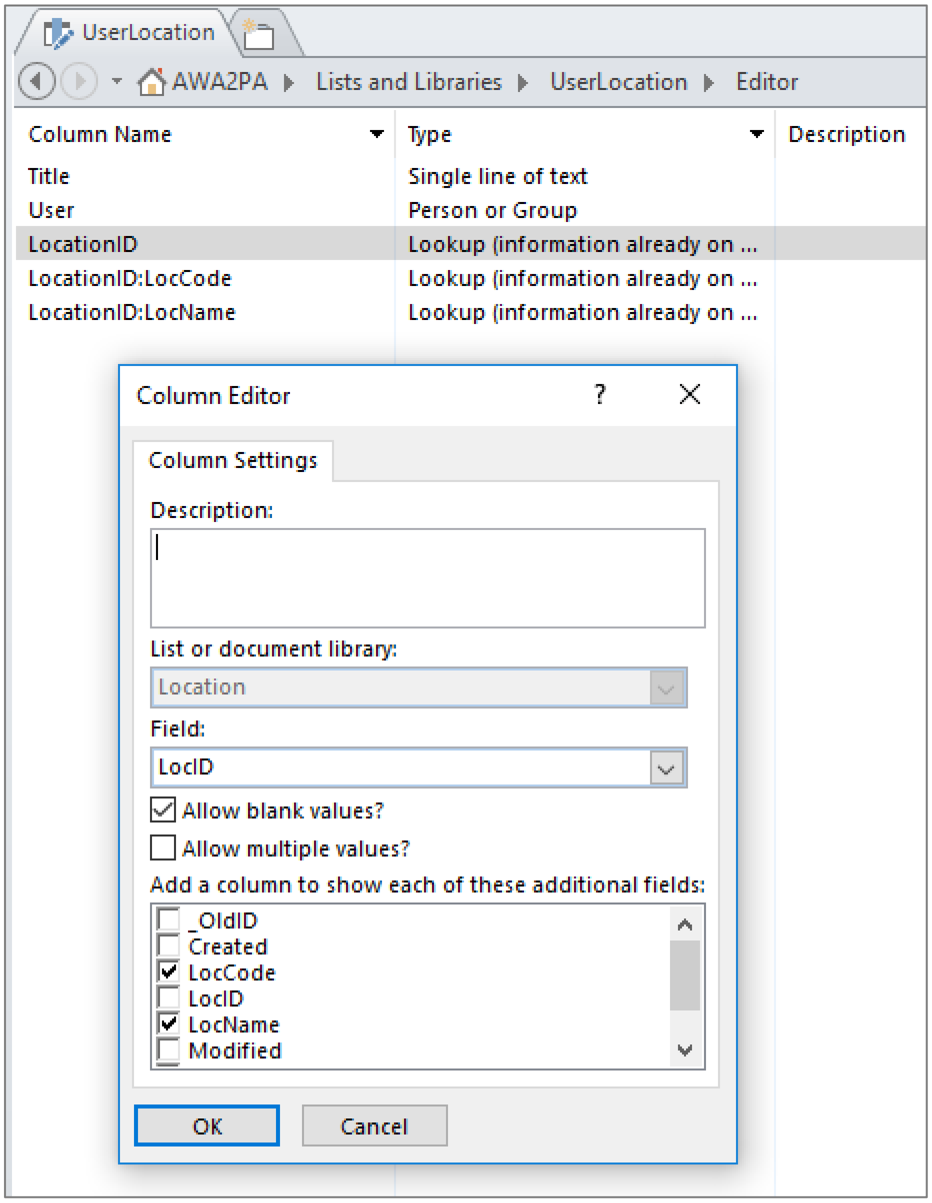
Approach two: using the SharePoint lookup field

This solution assumes you are using SharePoint which can include additional fields from lookup fields. The advantage with this approach is that it decreases the requirement for creating formulas within PowerApps. But the disadvantage is the expense of the additional work required to modify the list structure. When you create the list, it does not have a Lookup field, and SharePoint does not permit you to convert an existing field into a Lookup field. In this situation, you might find it preferable to create a blank SharePoint list with the schema you want. For instance, you can use SharePoint Designer to quickly design a blank list as the following example shows:



Name	Type
Lists	
Customer	Lists
Cylinder	Lists
ItemCode	Lists
Location	Lists
MicroFeed	Lists
Tracking	Lists
UserLocation	Lists
Document Libraries	
Documents	Document ...
Site Assets	Document ...
Site Pages	Document ...

You can use Access to load the data from the original Access web apps table to the modified SharePoint list, populating only the actual ID of the lookup field. Once it's populated, you can modify the list to include additional columns from the lookup field, as shown below:


UserLocation

AWA2PA ▸ Lists and Libraries ▸ UserLocation ▸ Editor

Column Name	Type	Description
Title	Single line of text	
User	Person or Group	
LocationID	Lookup (information already on ...	
LocationID:LocCode	Lookup (information already on ...	
LocationID:LocName	Lookup (information already on ...	

Column Editor

Column Settings

Description:

List or document library: Location

Field: LocID

☑ Allow blank values?

☐ Allow multiple values?

Add a column to show each of these additional fields:

- ☐ _OldID
- ☐ Created
- ☑ LocCode
- ☐ LocID
- ☑ LocName
- ☐ Modified

In PowerApps, you can then use this formula for the value:

Tracking.CylinderNumber

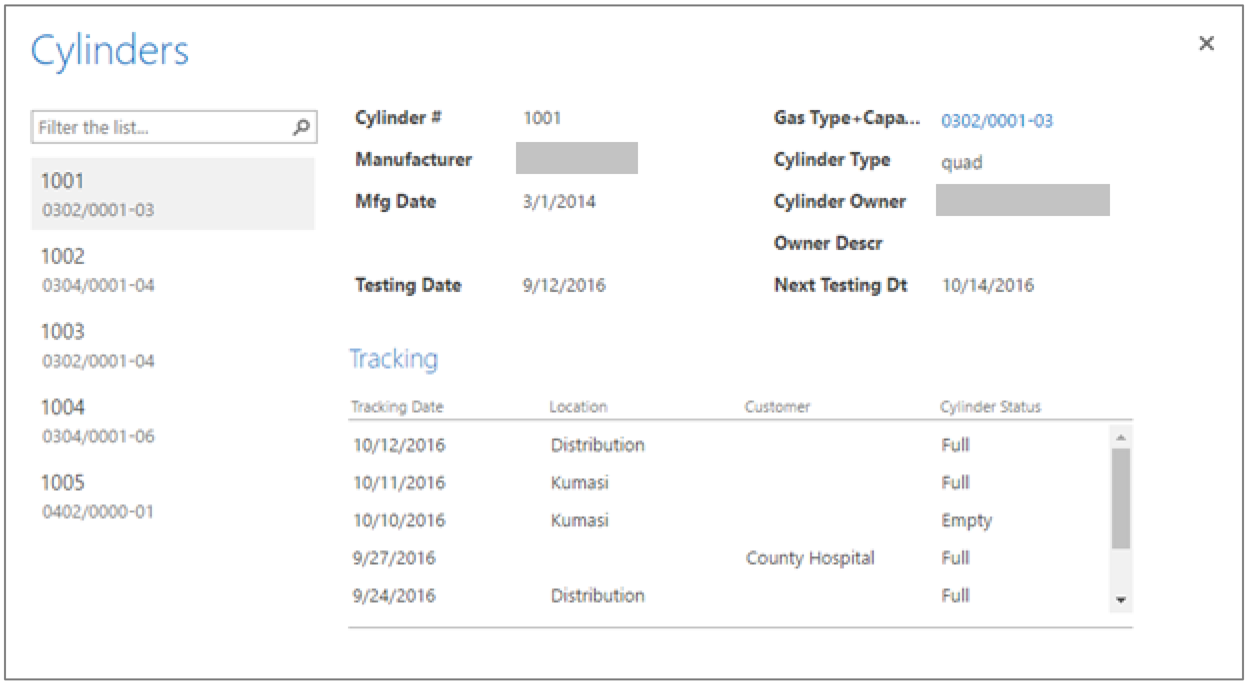
Note that up to now, the examples have considered only the phone layout. If you opt to use tablet layout, you might want to place multiple forms on the screen, and use the Selected property to achieve the relationship between master and detail forms or to conveniently search for data, which is discussed in the next section.

Searching and filtering

Many apps in an organizational environment require some search or filter features. Two important considerations are the app layout and the search controls configuration.

Layout perspective

When using a phone layout, it is usually preferable that there is only one form per screen. So, if you had an Access web apps list form where you had a sidebar for listing and searching with a detail area, that sidebar won't replicate well to PowerApps in the phone layout.



Cylinders

Filter the list...

1001
0302/0001-03

1002
0304/0001-04

1003
0302/0001-04

1004
0304/0001-06

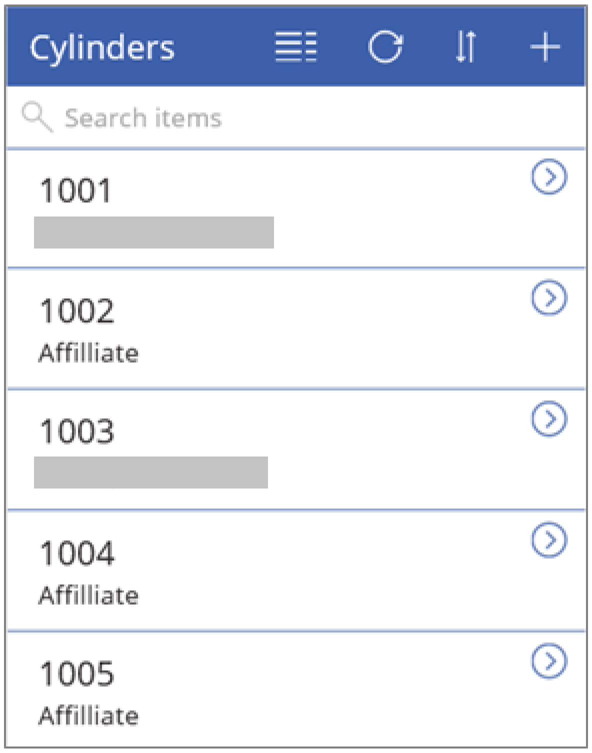
1005
0402/0000-01

Cylinder #	1001	**Gas Type+Capa...**	0302/0001-03
Manufacturer	[redacted]	**Cylinder Type**	quad
Mfg Date	3/1/2014	**Cylinder Owner**	[redacted]
		Owner Descr	
Testing Date	9/12/2016	**Next Testing Dt**	10/14/2016

Tracking

Tracking Date	Location	Customer	Cylinder Status
10/12/2016	Distribution		Full
10/11/2016	Kumasi		Full
10/10/2016	Kumasi		Empty
9/27/2016		County Hospital	Full
9/24/2016	Distribution		Full

In this case, you can create two screens, one to represent the list box for listing and searching (as the following example shows), and another to provide the details (not shown):



1001
[redacted]

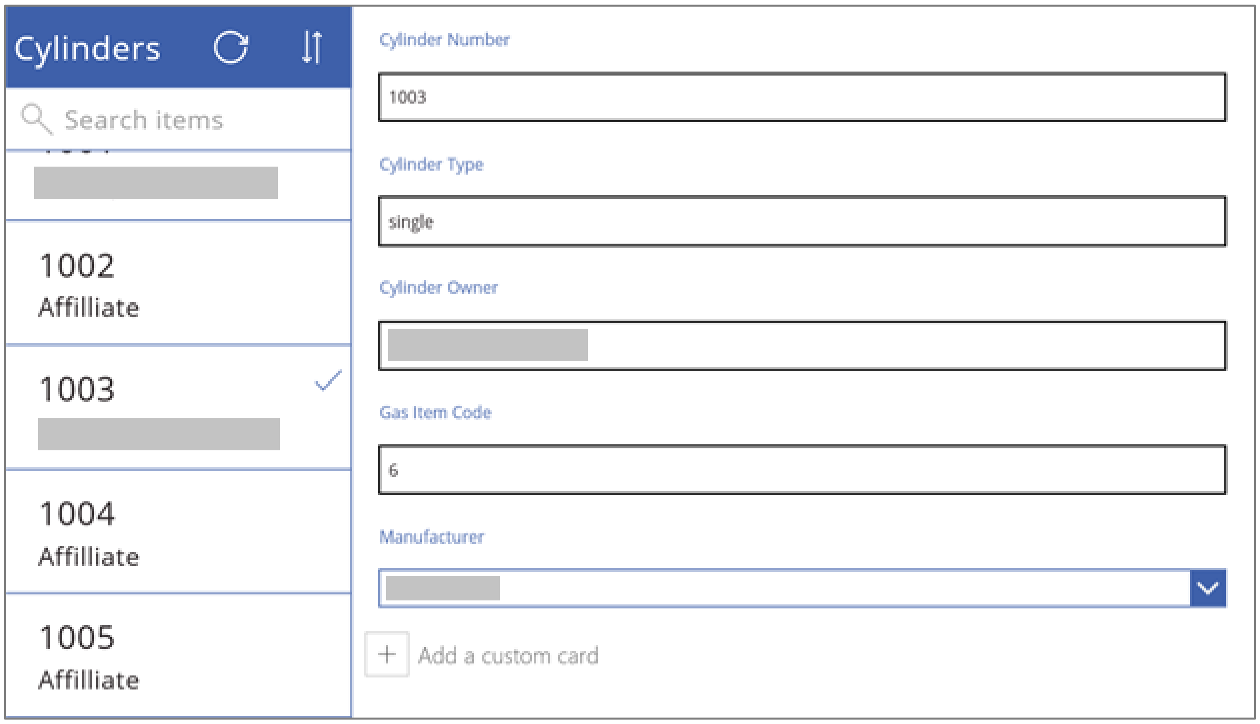
1002
Affilliate

1003
[redacted]

1004
Affilliate

1005
Affilliate

Alternatively, in a tablet layout, you can retain the same layout as an Access web apps list form, by placing two forms on a single screen.



[redacted]

1002
Affilliate

1003
[redacted]

1004
Affilliate

1005
Affilliate

Cylinder Number
1003

Cylinder Type
single

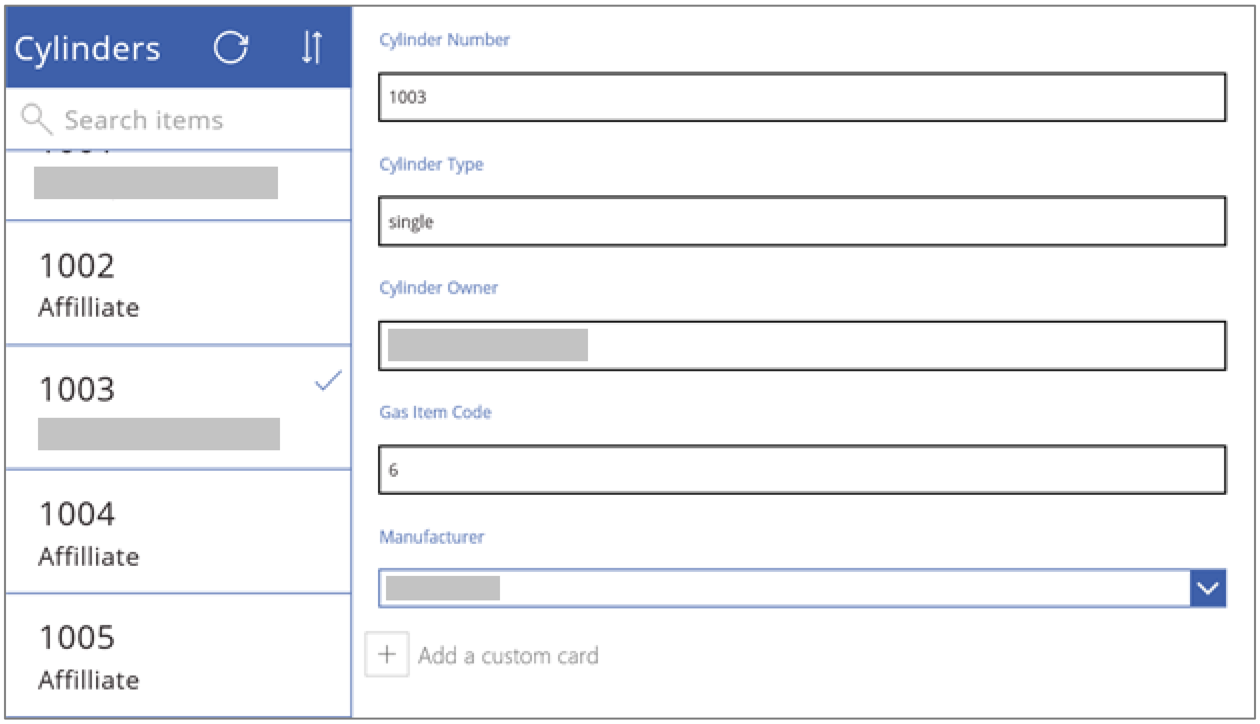
Cylinder Owner
[redacted]

Gas Item Code
6

Manufacturer
[redacted]

PowerApps gives you more control over how you can lay out forms in a screen. In this layout, you have one form on the left for listing and searching, and another form on the right to give more detail. But you don't have to have just two forms. You could further split into three forms or more. Although you might find a bit more work is required to get the functionality of the Access web apps list form, you will appreciate the extra freedom PowerApps gives you in customizing the forms.

Unlike Access web apps, there is no built-in linking between a form and related items control; in fact, there is no related items control at all in PowerApps. However, this is not a big loss and a great example of why PowerApps formulas are so flexible. Whether you do this with a phone or tablet layout, the formula remains the same in either layout. On the form for details, you could set the Item property to the selected value of the form for searching and listing. That does not require you to know the details of the data source's primary key as the Access web apps related items control linking usually does.



[redacted]

1002
Affilliate

1003 ✓
[redacted]

1004
Affilliate

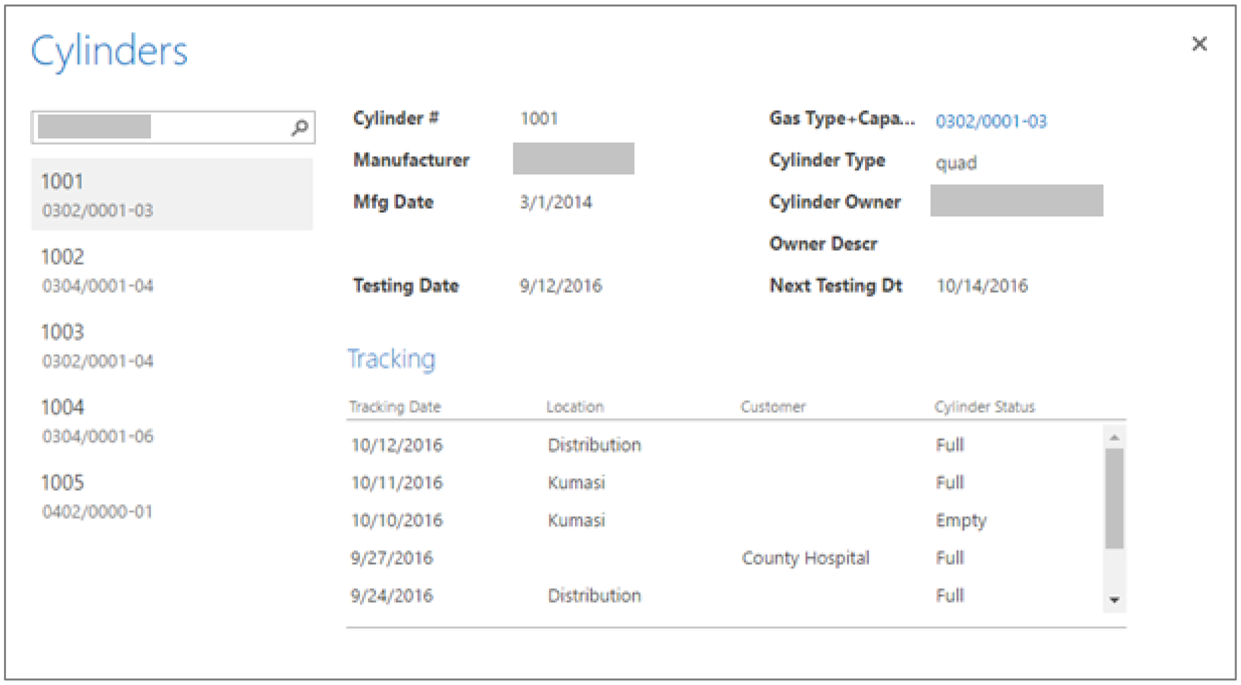
1005
Affilliate

Cylinder Number	1003
Cylinder Type	single
Cylinder Owner	[redacted]
Gas Item Code	6
Manufacturer	[redacted]

Search controls

Note This section was written before the Search function was added to PowerApps. The Search formula is much easier to implement than a nested Filter formula. For more information, see Filter, Search, and LookUp functions in PowerApps.

Configuring a control for searching, however, is not as simple. Let's start small and expand. On an Access web apps List form, you can type in anything and it matches any content of any field in the form's record source. This provides a search experience like popular web search engines where you only have one place to search, as opposed to a more traditional approach of creating different controls for each column to search on.



Cylinders

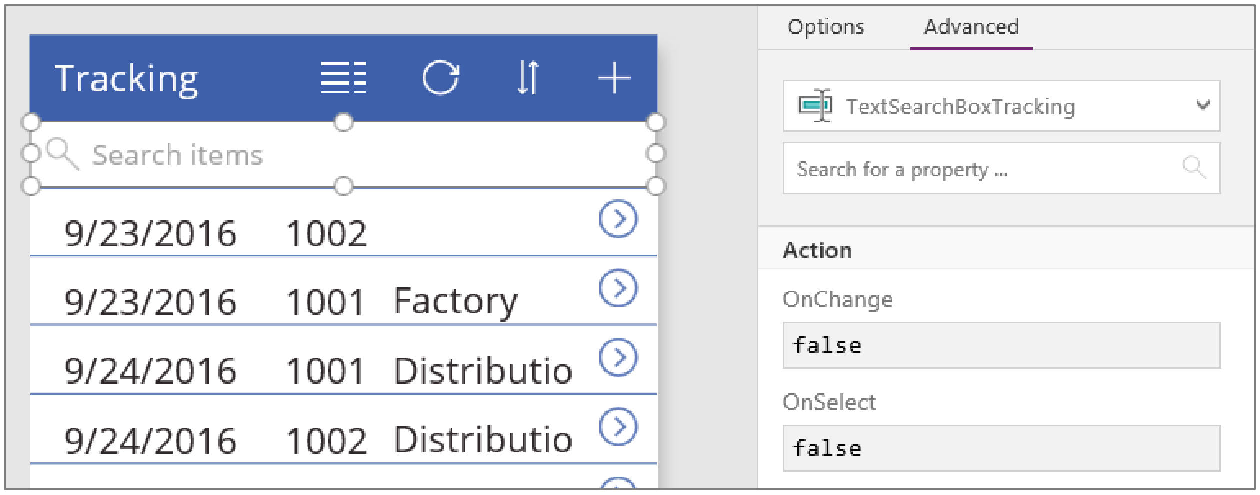
Cylinder #	1001	Gas Type+Capa...	0302/0001-03
Manufacturer		Cylinder Type	quad
Mfg Date	3/1/2014	Cylinder Owner	
		Owner Descr	
Testing Date	9/12/2016	Next Testing Dt	10/14/2016

1001 0302/0001-03
1002 0304/0001-04
1003 0302/0001-04
1004 0304/0001-06
1005 0402/0000-01

Tracking

Tracking Date	Location	Customer	Cylinder Status
10/12/2016	Distribution		Full
10/11/2016	Kumasi		Full
10/10/2016	Kumasi		Empty
9/27/2016		County Hospital	Full
9/24/2016	Distribution		Full

To replicate this functionality in PowerApps, you can use nested expressions. Coming from a background of using an Access desktop database or Access web apps, it might be intuitive to start writing the logic on the search button as an event, such as the AfterUpdate event on a search text box. But in PowerApps, you don't do that at all.



9/23/2016	1002	
9/23/2016	1001	Factory
9/24/2016	1001	Distributio
9/24/2016	1002	Distributio

OnChange: `false`

OnSelect: `false`

That's because with PowerApps, you do not use event-driven programming, but rather use formulas which are in effect always "on". For example, as a user types in a new search term, all formulas dependent on that value are automatically recalculated. This behavior is similar the automatic recalculation of an Excel workbook. In Excel, you don't need to write change event code to update other cells. You just define the dependency between the cells with formulas and Excel takes care of the rest.

This automatic recalculation means you set up the form's Items property with a formula. The major challenge here is that the formula must contain nested expressions which can cause it to be somewhat difficult to read. Note that the following Filter formula example uses the *in clause* which is not currently supported in a SharePoint list. To create these formulas, start with logic for only one column, without any sorting functionality.

Filter(Tracking, Or(IsBlank(TextSearchBoxTracking.Text), TextSearchBoxTracking.Text in Text(TrackingDate)))

You can use the Filter function and apply it to the data source "Tracking". However, you don't want to filter anything if the search text box is blank. It is necessary to have an Or function in the formula parameter of the Filter function. Note that the following syntax is also valid but won't work:

Filter(Tracking, IsBlank(TextSearchBoxTracking.Text), TextSearchBoxTracking.Text in Text(TrackingDate))

The Filter function permits multiple Formula parameters, but the catch is that by default, they all are applied with And logic, not Or logic. For that reason, you should be careful to ensure that you nest the optional path accordingly.

At this point, you should test and verify that the formula works as expected before you add another formula parameter to the Filter. Since you want to search on multiple columns, you don't want to exclude any results using the And logic. Therefore, add the new formula to the Or function, rather than directly to the Filter function, as the following example shows:

Filter(Tracking, Or(IsBlank(TextSearchBoxTracking.Text), TextSearchBoxTracking.Text in Text(TrackingDate), CylinderID = LookUp(Cylinder, CylinderNumber = TextSearchBoxTracking.Text).ID))

The LookUp function is necessary because the Tracking data source does not contain the CylinderNumber, even though you display CylinderNumber using the LookUp function.

So, even though the form has the cylinder numbers, users will want to type cylinder numbers, not the system ID used by the database. The assumption is that the CylinderNumber is not added as a lookup field as discussed in the previous section. In that case, you can do an additional lookup of the Cylinder data source to compare the ID and thus search for the Tracking data source with that ID representing the CylinderNumber entered by the user. Note that the ID property references the result of the LookUp function on the Cylinder data source.

> *In the list, the control was originally called "CylinderID", and Access web apps displays that column as "CylinderID". However, PowerApps does not have user-friendly labels and might use programmatic names such as "ID" or "Title". As a rule, when using a custom list, the programmatic name is the same as the original display name, and never changes. For example, if you created a new SharePoint list and added a column "My Rating", the programmatic name is "My_x0020_Rating", and remains that way even if you later change it to "# of Stars given".*

The last example to cover is when you need to depend on another control. As shown in the screenshot, there is a slider control that indicates whether to filter by current user's location or show all records. Here's the formula:

Filter(Tracking, Or(IsBlank(TextSearchBoxTracking.Text), TextSearchBoxTracking.Text in Text(TrackingDate), CylinderID = LookUp(Cylinder, CylinderNumber = TextSearchBoxTracking.Text).ID), Or(ToggleFilterByUserLocation.Value = false, LocationID = LookUp(UserLocation, UserLocation[@User].Email = User.Email, LocationID).Id))

This formula seems to be getting complicated very quickly. The first thing to note is that this formula is within a second formula parameter of the Filter function, and there is a second Or argument to represent that parameter:

In other words, Filter(Tracking, Or(…), Or(…)) is the same thing as If (A Or B) And (C Or D) Then.

Nested within the second Or argument, the logic is like the first column, which skips this part if the slider control is set to the off position. Otherwise, the formula evaluates whether the location of that cylinder matches the user's location. A lookup of the location's ID is needed to get the location name in the Tracking data source. The formula fetches this value from the UserLocation data source. However, be aware of this issue: within the UserLocation data source, there is a column named "User", but that's also a function in PowerApps which returns information about the currently logged in user. That could be ambiguous to PowerApps. To help PowerApps know which "User" you want, use the disambiguation operator "@", "UserLocation[@User]", which essentially means "User from UserLocation". Note the use of User.Email, which calls the PowerApps User function and returns the Email property.

The last touch to put on the formula is to support sorting operations. Wrap the entire Filter function in the SortByColumns function:

SortByColumns(Filter(Tracking, Or(IsBlank(TextSearchBoxTracking.Text), TextSearchBoxTracking.Text in Text(TrackingDate), CylinderID = LookUp(Cylinder, CylinderNumber = TextSearchBoxTracking.Text).ID), Or(ToggleFilterByUserLocation.Value = false, LocationID = LookUp(UserLocation, UserLocation[@User].Email = User.Email, LocationID).Id)), "TrackingDate", If(SortDescending1, Descending, Ascending))

In this case, sorting occurs for only one column. You can apply similar ideas from the filtering examples if you want to sort by multiple columns. However, this includes more function nesting and can make it a challenge to test. For that reason, it's a good idea to build up the nested functions from innermost to outermost nesting instead of constructing all the functions from left to right, which can be more problematic.

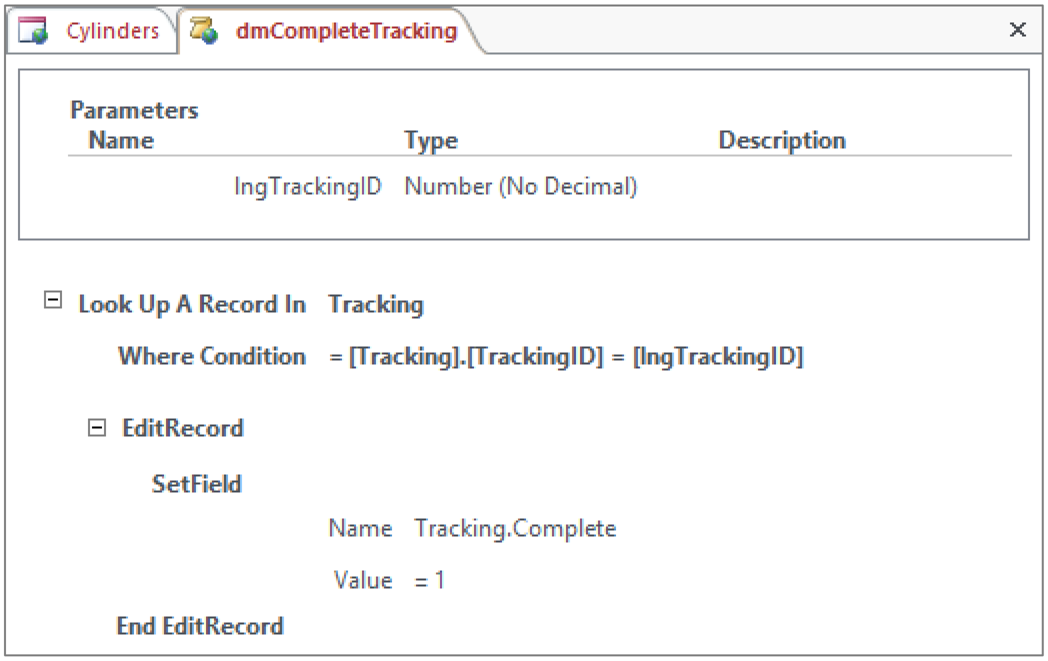
It's worth reviewing the section, Client-side vs server-side functionality, where the concept of delegation in PowerApps is discussed. To make it easier to know what is and is not being delegated, the authoring experience provides blue dot suggestions when a formula contains something that cannot be delegated. Blue dots are only shown on formulas that operate on delegable data sources. If you don't see a blue dot and you believe your formula is not being properly delegated, check the type of data source against the list of delegable data sources.

Data macros

A significant portion of business logic in Access web apps is found in data macros. There are also UI macros, but the focus of this section is data macros which are used to manipulate data directly. In Access web apps, a data macro is typically executed directly on the database server. In contrast, PowerApps does not usually execute any operation directly on the database but instead retrieves the data, works on the copy of that data, and then merges it back to the database.

The PowerApps team is working on making delegation available for more and more data sources so this limitation may be rectified. Even with delegation, it's still beneficial to be conservative with the scope of bulk updates. For that reason, you might discover not all data macros can be replicated from Access web apps to PowerApps. It might be more accurate to envision this as re-thinking your strategies because the approaches used are quite different.

Data macros are procedural, working on single record by record. By contrast, PowerApps formulas are declarative. To illustrate, consider a simple data macro that might be driven by a button to mark a record as complete:



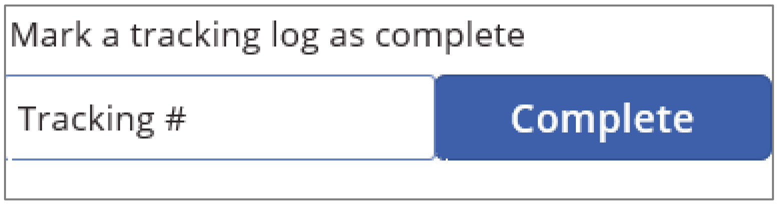
Parameters

Name	Type	Description
lngTrackingID	Number (No Decimal)	

```
Look Up A Record In   Tracking
     Where Condition  = [Tracking].[TrackingID] = [lngTrackingID]

   EditRecord
       SetField
                      Name   Tracking.Complete
                      Value  = 1
     End EditRecord
```

Note that a LookupRecord macro is performed before you can edit the record. This works for situations when the record is not currently loaded into the Access web apps form on which the button is located and so it can be used independently. Here's the equivalent user interface in PowerApps:



In this case, the update operation happens only when users select the button, so use the button's OnSelect action with the following formula:

Patch(Tracking, LookUp(Tracking, ID = Value(TrackingIDTextInput.Text)), {Complete:true})

Just like data macros, use a LookUp function to retrieve a single record in the second argument. In the third argument, modify the record by setting the Complete column to be true. Note the use of the Patch function. There is also a function named Update, but you must enumerate all columns in a record when assigning values to the Update function's new argument. The Patch function only changes what you want to change. For that reason, you'll likely find yourself using the Patch function more often.

You can also use the Patch function to perform bulk changes to data. Remember, you can only work with data that is already available locally in PowerApps, which is a subset of what may be in the data source.

Note that the last argument is wrapped in curly braces, which indicates that it is a record data type. In this example, only one column is updated. However, you are not limited to just one column; you can add other columns. For example, the following parameter:

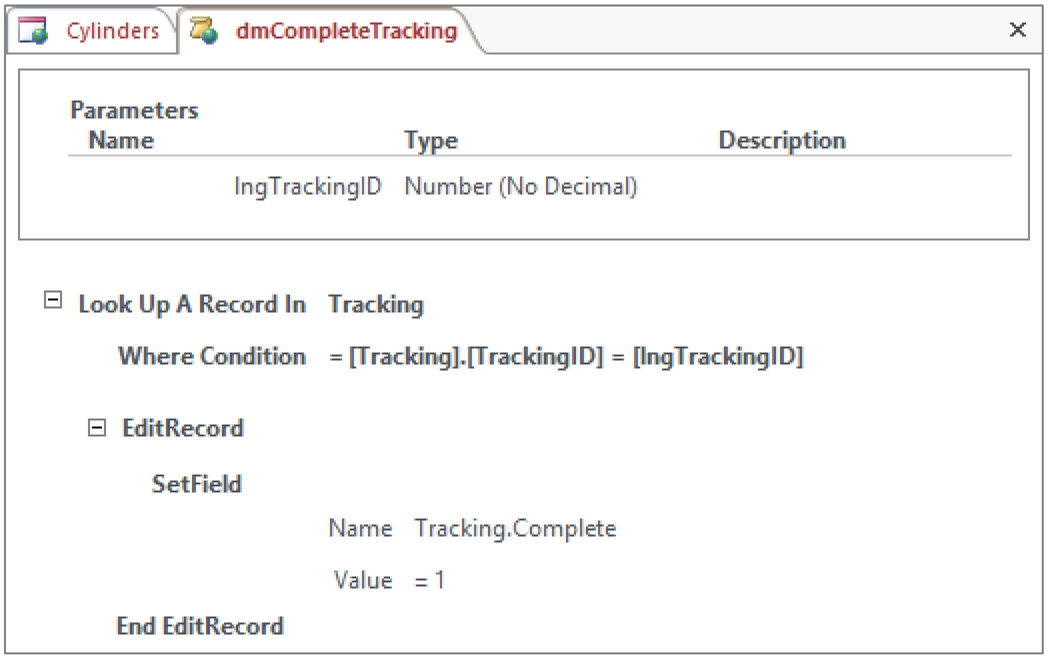
{Complete: true, Cystitis: 1}

Would update two columns in a single Patch function call.

Inserting records requires that you use the Defaults function to create a record with the default values. You can then modify the record's columns just like you would with an update that uses the Patch function to insert it into the data source. For example:

Patch(Tracking, Defaults(Tracking), {Complete: true})

Consider another data macro, a For Each Record block that loops over each record. For example, you want to mark all records as complete for a specified date at a given location.

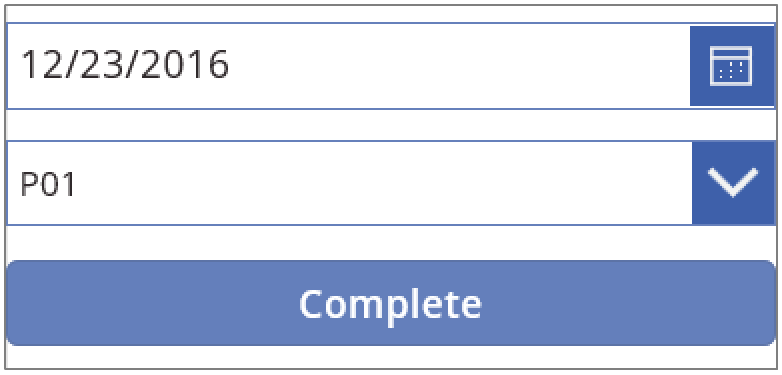


```
Parameters
 Name                Type                    Description
          lngTrackingID   Number (No Decimal)

⊟ Look Up A Record In   Tracking
     Where Condition    = [Tracking].[TrackingID] = [lngTrackingID]

   ⊟ EditRecord
        SetField
                      Name   Tracking.Complete
                      Value  = 1
     End EditRecord
```

Recall that with Access web apps, there is no support for action queries, so to do a bulk update requires the use of a For Each Record block. In an Access desktop database, you could use an Update query to do the same thing. In PowerApps, you can do something similar with formulas.



The following formula does the equivalent operation in PowerApps:

UpdateIf(Tracking, TrackingDate=TrackingDatePicker.SelectedDate && toLocation = LocationDropdown.Selected.ID && Complete = false, { Complete: true })

Because you want to update multiple records based on a condition, the UpdateIf function is a better fit than the Patch function. Note that unlike the formulas used for searching in the previous section, you can use multiple conditions expressed as a single argument.

Like the Patch function, multiple columns can be updated by adding the other columns in the record variable of the third argument. In the rare case when you do want to update all records unconditionally, you can accomplish this by returning true in the second argument.

When you need to delete records, you can use the Remove If function in a similar way to the UpdateIf function. You can also use the Clear function if you want to delete all records in a data source, or the Remove function if you have a table variable representing the records you want to remove from the data source. PowerApps also offers a For All function which is the equivalent of a ForEachRecord data block in Access web apps.

What you can do in Access web apps but not in PowerApps

Important Keep in mind that PowerApps is a new platform. More likely than not, some of the features discussed may change. For the latest information, see www.PowerApps.com.

Access web apps has capabilities that PowerApps currently does not have.

Inviting anonymous users

Access web apps allows you to invite users outside of SharePoint. All they need is a free Microsoft account. With PowerApps, the person you invite must be within your same domain and must have rights to the data source, such as a SharePoint List, that your app is based on.

Datasheet views

Although there is a potential work-around for datasheet views, this can be quite cumbersome for some users, especially when viewing many fields and having them sorted and filtered any way they want. Many users are comfortable with Excel and datasheet views and they need the exact look and feel, or they won't accept it.

Summary views

There are ways to filter data, use aggregate formulas, such as SUM() to add up total sales, for example. But, PowerApps can only filter for that client and shows all records for that client, not what their sales were per quarter, or other summaries. However, if you are using PowerApps, using Power BI might be a better fit overall for providing summarized data and some reporting capabilities which are more powerful than Access web apps summary views.

Transaction data tracking and troubleshooting

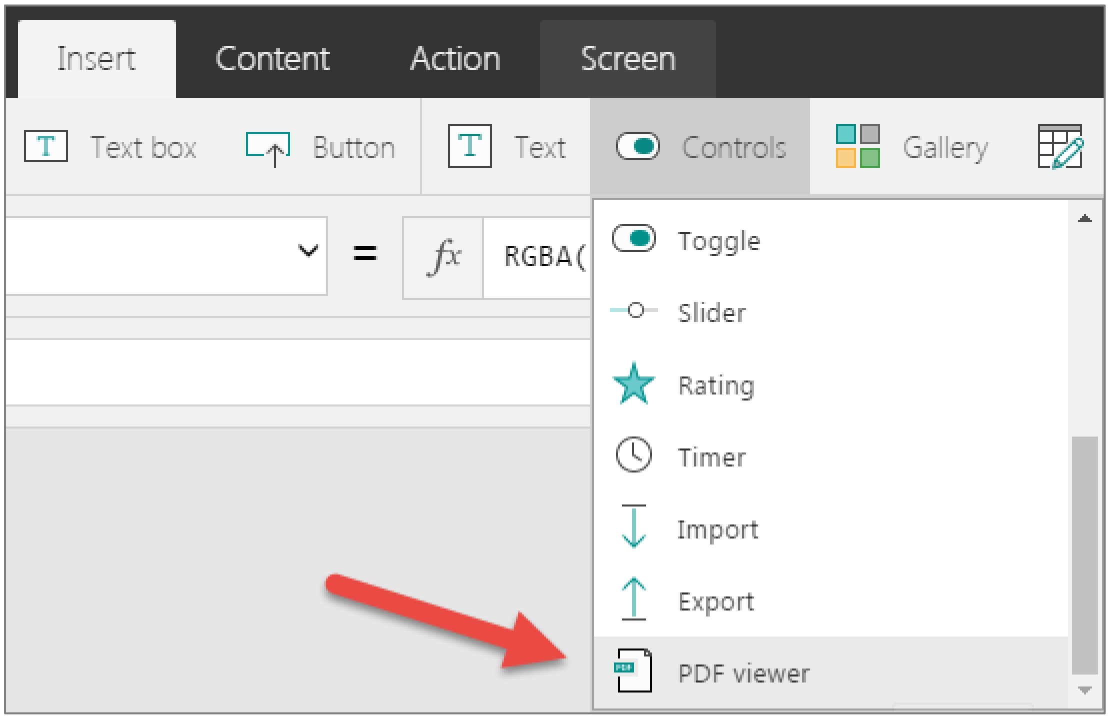
A little-known feature in Access web apps, which is very powerful for troubleshooting purposes, is the ability to set up tracing of data as a user works with the data.

What you can do in PowerApps but not in Access web apps

On the other hand, PowerApps has capabilities that Access web apps does not have.

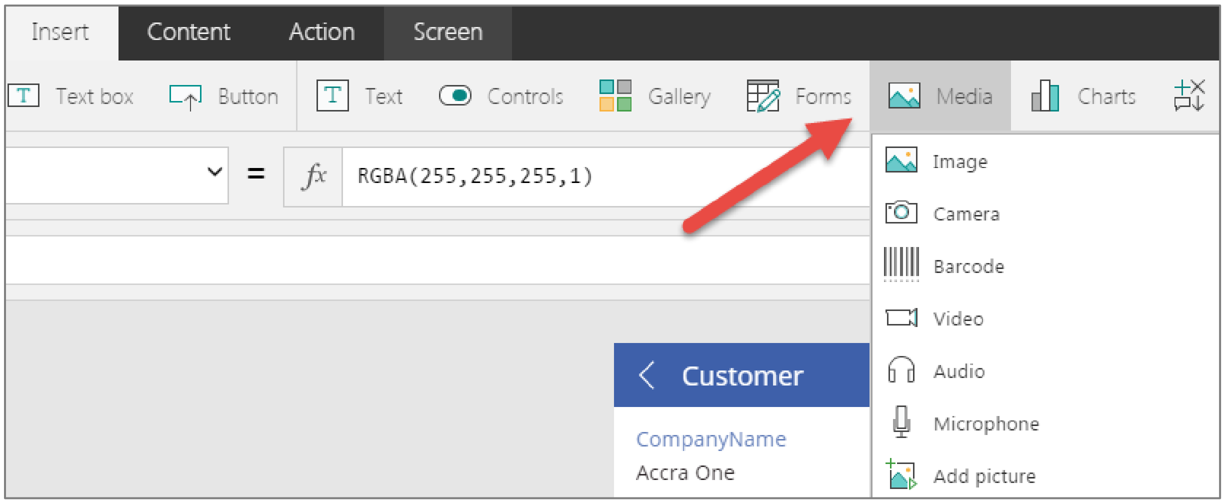
PDF viewer

PowerApps can view a PDF file from within the app. You can select this from **Insert** > **Controls**, and then scroll to the bottom where you see **PDF Viewer**.



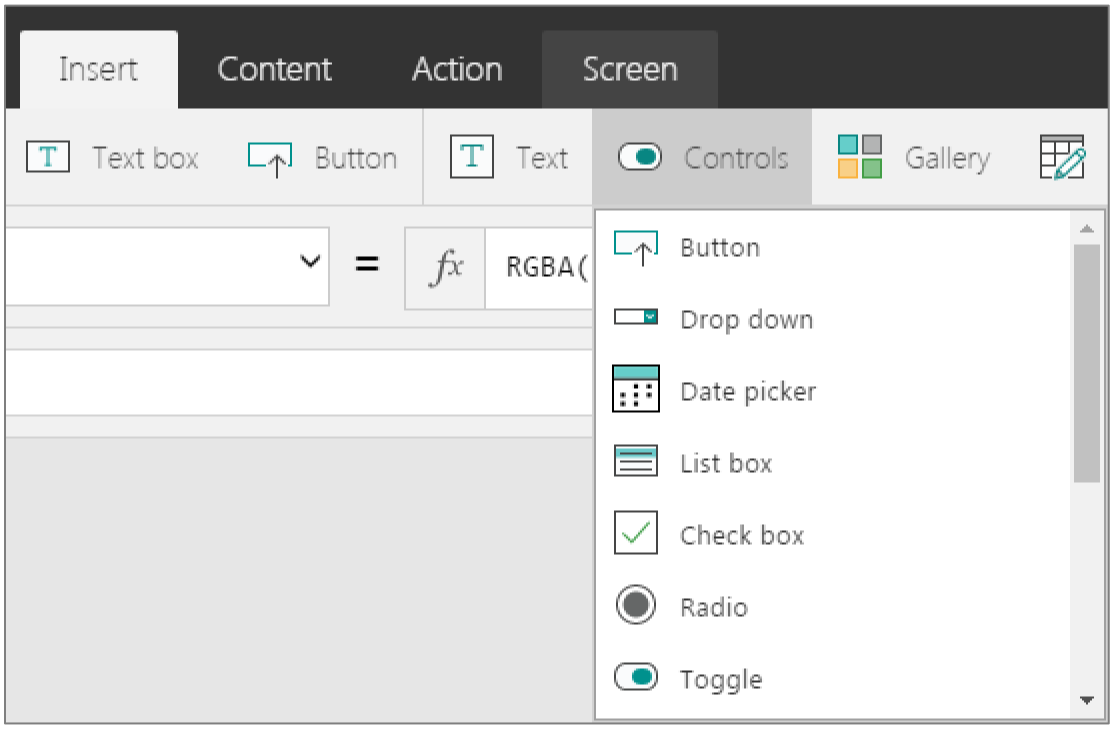
Multimedia support

PowerApps can include images, video, and audio within your app. To insert the proper control, go to **Insert** > **Media**, and then select the type of media. You also have access to the capabilities of a device, such as using a camera or microphone to capture images, video, and audio.



Variety of controls

PowerApps has more controls to choose. For example, select **Insert** > **Controls**.

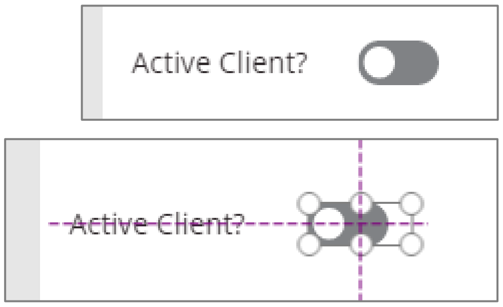


Toggle

The Toggle control turns something on or off.

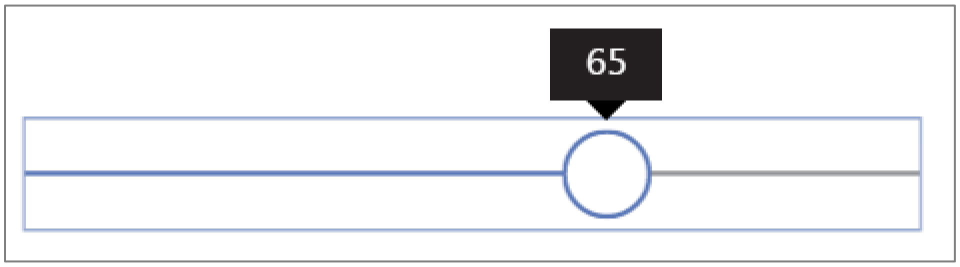
When you place the control, it lets you center it based on your **Text** (Label).

This ability is useful when you are focused on providing a great user experience.



Slider

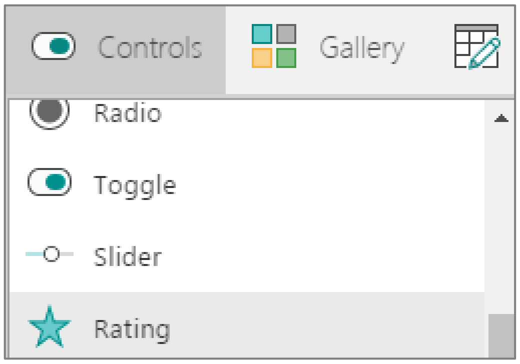
There is a Slider control, that is often used to select data such as cost (for buying a car), size (for selecting a waist size), car size selection (for rental cars, the different categories from compact to luxury), and so on.



On mobile devices, sliders are much better than drop-down menus.

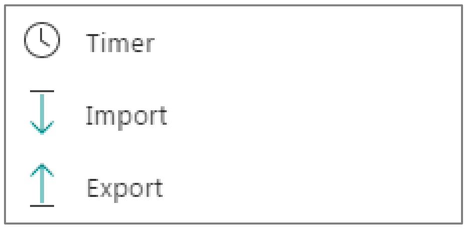
Rating

If you've ever installed an iPhone or Android app, you'll be familiar with the Rating control. This is also useful for conducting internal surveys.



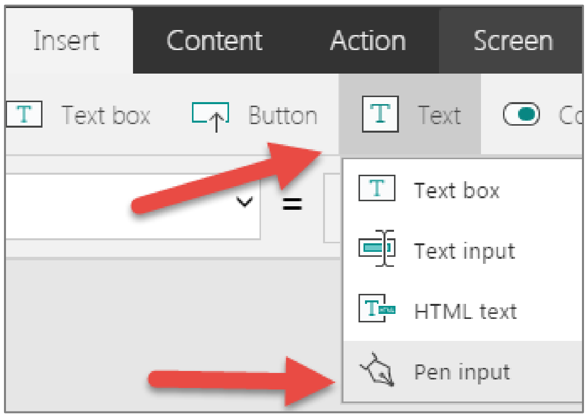
Other controls

There are also controls for adding a Timer, doing an Import operation, and doing an Export operation.



Pen input

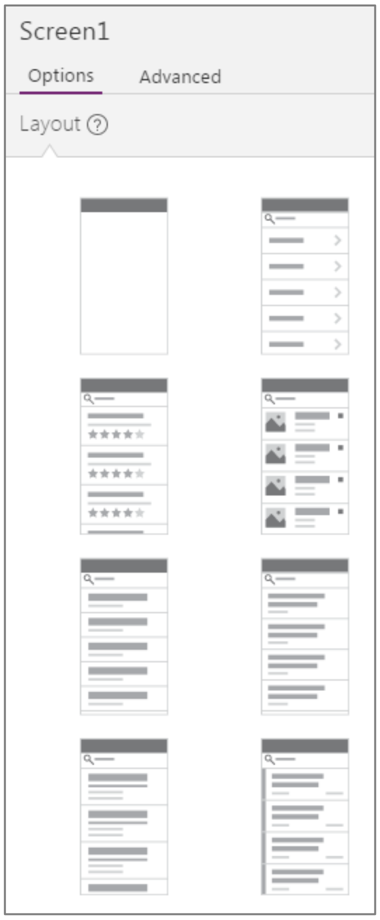
Pen input is becoming increasingly useful on a mobile device and helps improve the data entry methods in the user interface.



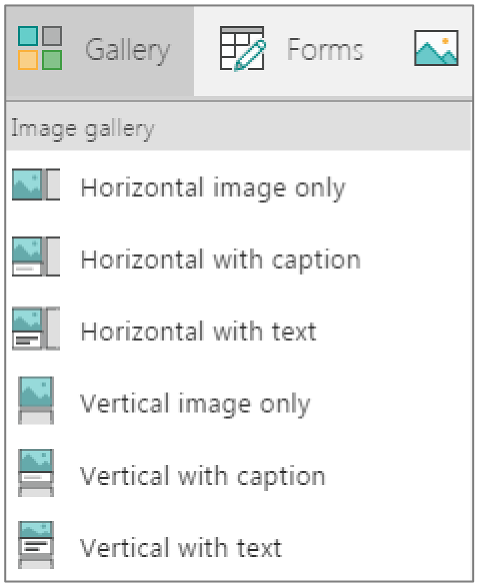
Variety of layouts and galleries

PowerApps has many options to arrange content compared to Access web apps. For example:

Layouts

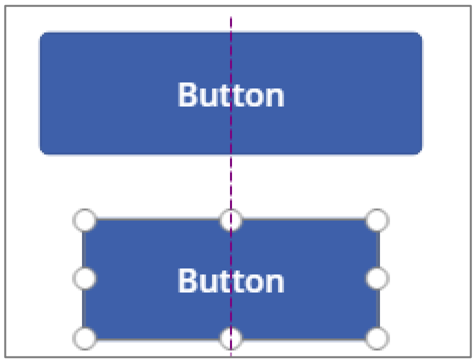


Galleries



Rich user interface building capabilities

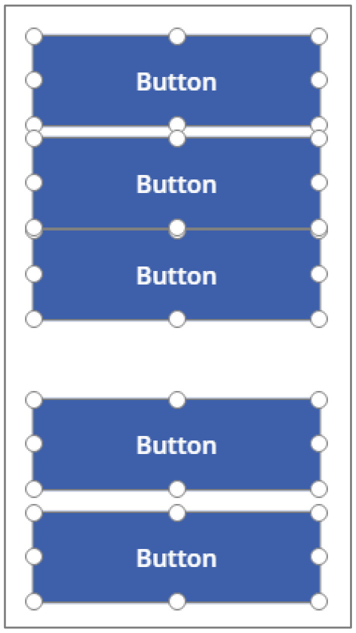
You can position and align PowerApps controls exactly where you want them on a screen, just like PowerPoint controls. You can even overlap controls and arrange a certain control to be on top. For example, you can easily align controls to the center:

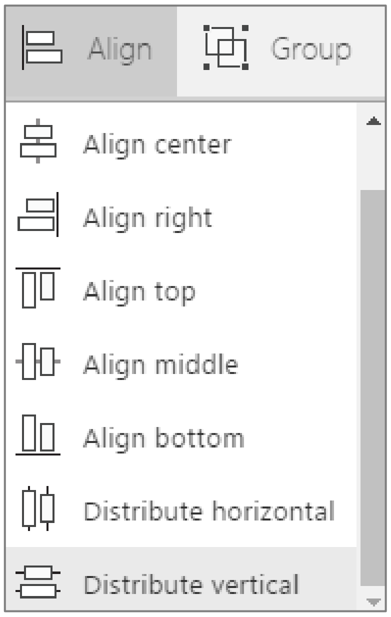


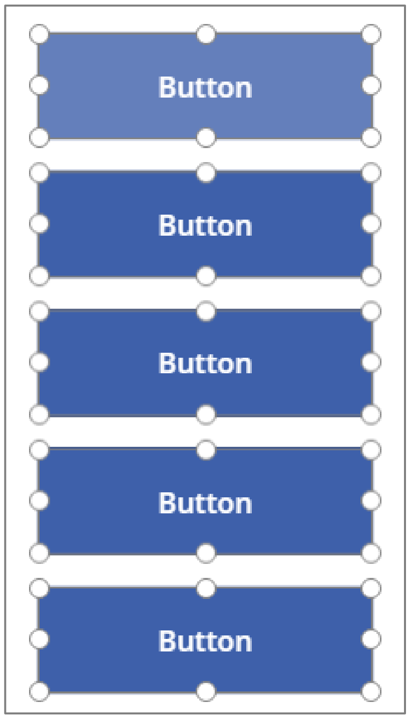
Or equally space controls in a vertical way:

1. Select all the buttons.
2. Choose **Distribute vertical**.

Result: buttons are evenly spaced.

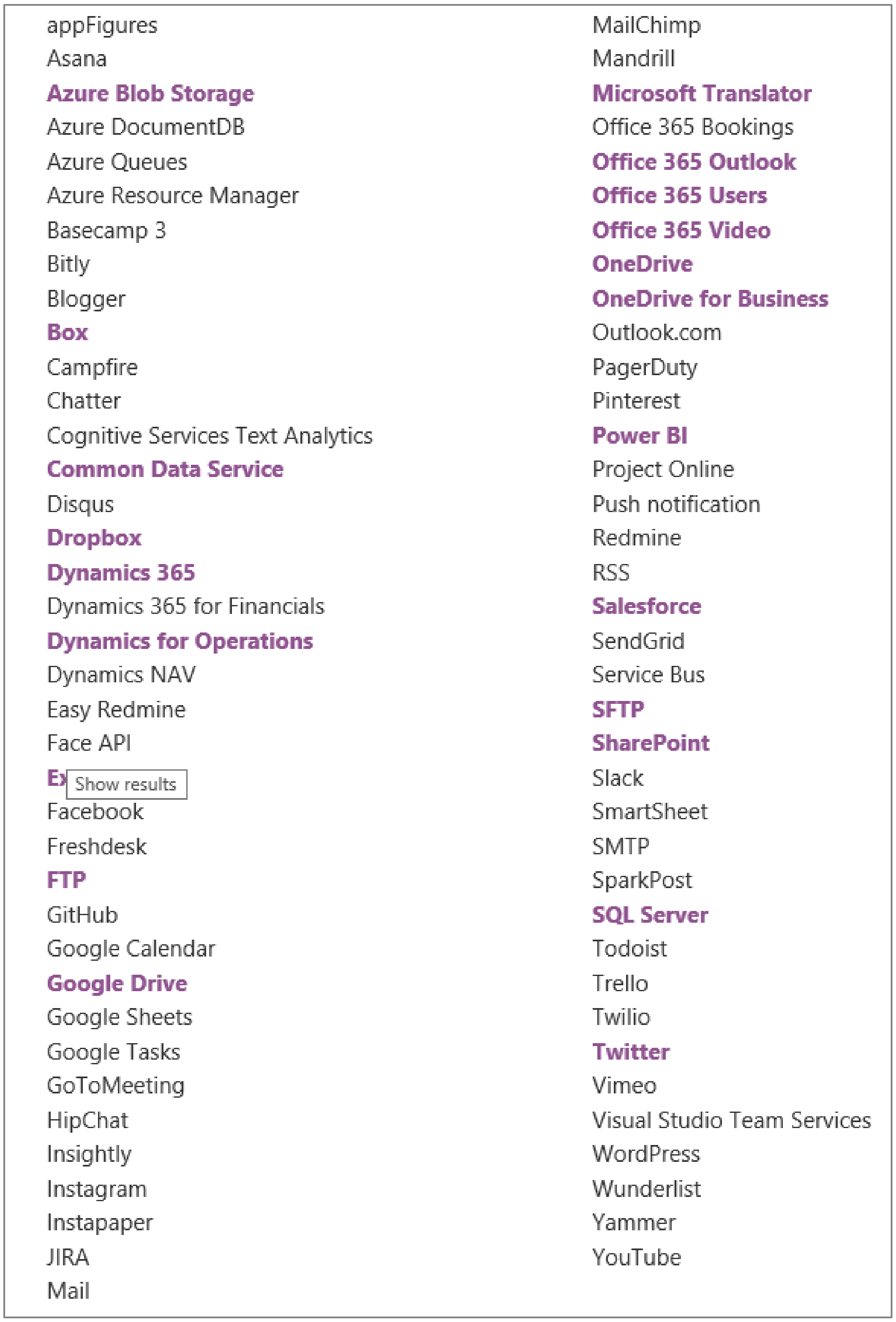






SharePoint is not the only data source

Although our example apps have used SharePoint lists as the data source, you are not limited to just SharePoint lists. There are dozens of data sources available, and more are being added.



appFigures
Asana
Azure Blob Storage
Azure DocumentDB
Azure Queues
Azure Resource Manager
Basecamp 3
Bitly
Blogger
Box
Campfire
Chatter
Cognitive Services Text Analytics
Common Data Service
Disqus
Dropbox
Dynamics 365
Dynamics 365 for Financials
Dynamics for Operations
Dynamics NAV
Easy Redmine
Face API
Ex Show results
Facebook
Freshdesk
FTP
GitHub
Google Calendar
Google Drive
Google Sheets
Google Tasks
GoToMeeting
HipChat
Insightly
Instagram
Instapaper
JIRA
Mail
MailChimp
Mandrill
Microsoft Translator
Office 365 Bookings
Office 365 Outlook
Office 365 Users
Office 365 Video
OneDrive
OneDrive for Business
Outlook.com
PagerDuty
Pinterest
Power BI
Project Online
Push notification
Redmine
RSS
Salesforce
SendGrid
Service Bus
SFTP
SharePoint
Slack
SmartSheet
SMTP
SparkPost
SQL Server
Todoist
Trello
Twilio
Twitter
Vimeo
Visual Studio Team Services
WordPress
Wunderlist
Yammer
YouTube

Extensibility

Lastly, PowerApps has more extensibility than Access web apps. Access web apps do provide a way to enhance the interface using Microsoft Office Web Apps. But PowerApps can use Microsoft Flow, to create an InfoPath-like process, and APIs to increase functionality programmatically.

About the authors

As Microsoft MVPs for Access, Ben Clothier (consultant and well-known author of Access books) and Andy Tabisz (consultant, national speaker, and Access Web Apps User Group president) are providing this introduction to PowerApps from the perspective of an Access web apps user and developer.