
Canvas apps embedding SDK in PowerApps

Public Preview, Version 0.3.0

Copyright

This document is provided "as-is". Information and views expressed in this document, including URL and other internet web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. This document is confidential and proprietary to Microsoft. It is disclosed and can be used only pursuant to a non-disclosure agreement.

© 2019 Microsoft. All rights reserved.

Microsoft is a trademark of the Microsoft group of companies. All other trademarks are the property of their respective owners.

Contents

Conceptual Overview	8
Canonical Scenarios	8
<i>Enabling makers to build standalone apps on your services</i>	<i>8</i>
<i>Enabling makers to customize and extend your app</i>	<i>10</i>
<i>Enabling a maker extension ecosystem for your application</i>	<i>12</i>
<i>Hosting and Licensing Considerations</i>	<i>13</i>
<i>PowerApps hosted in your customers tenant</i>	<i>14</i>
<i>PowerApps hosted in your tenant</i>	<i>14</i>
Quick Start Scenarios	14
Launching PowerApps Studio	14
Embedding a canvas app	15
Canvas app embedding SDKs	16
PowerApps Authoring SDK	16
Classes.....	16
Interfaces	16
Global Methods	17
Enums	17
Types.....	17
Global Methods	18
initAsync	18
<i>Syntax</i>	<i>18</i>
<i>Parameters</i>	<i>18</i>
<i>Return Value</i>	<i>18</i>
<i>Examples.....</i>	<i>18</i>
Classes	18
MakerSession	18
<i>Syntax</i>	<i>19</i>
<i>Methods.....</i>	<i>19</i>
<i>Event.....</i>	<i>20</i>
EditAppAsync.....	20
<i>Syntax</i>	<i>20</i>
<i>Parameters</i>	<i>20</i>
<i>Return Value</i>	<i>20</i>

CreateAppAsync	20
Syntax	20
Parameters	20
Return Value	21
CreateAppWithSqlConnectionAsync.....	21
Syntax	21
Parameters	21
Return Value	21
SetSchemaAsync.....	21
Syntax	21
Parameters	21
Return Value	22
SetDataAsync.....	22
Syntax	22
Parameters	22
Return Value	22
SetHostMethodDefinitionAsync.....	22
Syntax	22
Parameters	22
Return Value	22
disposeAsync	23
Syntax	23
Parameters	23
Return Value	23
Examples.....	23
appSaved	23
Syntax	23
Example	24
appPublished	24
Syntax	24
Example	24
StandardTemplate	25
Syntax	25
Methods.....	25
Constructor	25
Syntax	25
Parameters	25

Return Value	25
Examples.....	26
getTemplateName.....	26
Syntax	26
Return Value	26
Examples.....	26
EventHook	26
Methods.....	26
subscribe / unsubscribe / unsubscribeAll.....	27
Syntax	27
Parameters	27
Return Value	27
Example	27
Interfaces.....	27
SdkInitializer	27
Syntax	27
Properties	28
MakerSessionAppInfo.....	28
Properties	28
EditAppOptions	28
CreateAppOptions	28
CreateAppWithSqlConnectionOptions	29
PowerAppsSchema.....	31
Enums	31
VerticalLayouts	31
HorizontalLayouts.....	33
FlexibleLayouts	34
FormFactor	36
Types.....	36
EventListener.....	36
Definition	36
Template.....	36
Definition	37
Layout.....	37
Definition	37
PowerApps Player SDK	37
Classes.....	37

Interfaces	37
Global Methods	38
Enums	38
Types.....	38
Global Methods.....	39
initAsync	39
Syntax	39
Parameters	39
Return Value	39
Examples.....	39
isSdkInitialized	39
Syntax	39
Parameters	39
Return Value	40
Examples.....	40
Classes	40
Player	40
Syntax	40
Methods.....	40
Event.....	41
initPlayerByAppId.....	41
Syntax	41
Parameters	41
How to get AppId.....	41
Return Value	42
Examples.....	42
initPlayerByLogicalName	42
Syntax	42
Parameters	43
How to get LogicalName, EnvironmentId, TenantId.....	43
Return Value	44
Examples.....	44
renderApp	44
Syntax	44
Parameters	44
Return Value	44
Examples.....	44

dispose.....	45
<i>Syntax</i>	45
<i>Parameters</i>	45
<i>Return Value</i>	45
<i>Examples</i>	45
setData	45
<i>Syntax</i>	45
<i>Parameters</i>	45
<i>Return Value</i>	45
<i>Examples</i>	46
registerHostCallback.....	46
<i>Syntax</i>	46
<i>Parameters</i>	46
<i>Return Value</i>	46
<i>Examples</i>	46
onAppLoaded	47
<i>Syntax</i>	47
<i>Example</i>	47
onAppUnload.....	47
<i>Syntax</i>	47
<i>Example</i>	47
onAppError	48
<i>Syntax</i>	48
<i>Example</i>	48
EventHook	48
<i>Methods</i>	49
subscribe / unsubscribe / unsubscribeAll.....	49
<i>Syntax</i>	49
<i>Parameters</i>	49
<i>Return Value</i>	49
<i>Example</i>	49
Interfaces.....	50
SdkInitializer	50
<i>Syntax</i>	50
<i>Properties</i>	50
RGBA.....	50
<i>Syntax</i>	50

<i>Properties</i>	50
PlayerOptions	51
<i>Syntax</i>	51
<i>Properties</i>	51
PlayerInfo	52
<i>Properties</i>	52
Enums	52
SdkState	52
Types	52
AccessTokenProvider	52
<i>Definition</i>	52
RecordData	53
<i>Definition</i>	53
TableData	53
<i>Definition</i>	53
PowerAppsData	53
<i>Definition</i>	53
Existing Docs	53
PowerApps	53
Power BI	53
Flow	53

Conceptual Overview

Now more than ever, businesses and organizations have embraced the value of using data to drive outcomes. They need software that is driven by data and powered by AI and cognitive services to help them fundamentally reimagine how they engage customers, empower employees, transform operations, and reimagine products and services. There are three pillars for how organizations should approach transforming their business with a digital feedback loop:

- **Buying off-the-shelf packaged applications** is the fastest way for your customers to transform their business and gets them 80% of the results.
- No application meets the exact criteria what the customer needs, the next wave of results come from **customizing those applications** to fit your customers' processes and industry.
- Sometimes there is no app and **creating a custom app** that perfectly fits into your customers' workflows are required to get the benefits of the last mile of digital transformation.

Microsoft PowerApps is high productivity, a low-code development platform that empowers app makers and enables anyone to build rich experiences. By embedding canvas apps in your application, you can quickly and easily transform your customers into makers. Your new makers can extend your applications with experiences ranging from creating simple custom forms to adding feature rich screens and everything in between. They can connect to data, create business logic, and orchestrate workflows all from within your application. By enabling your customers to act as makers, your applications and services can finally cross that last mile and fit perfectly into your customers' workflows making them even more valuable and indispensable.

Note: This feature is supported only in canvas apps.

Important:

- This is a preview feature.
- Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
- Microsoft doesn't provide support for this preview feature. Microsoft Technical Support won't be able to help you with issues or questions.

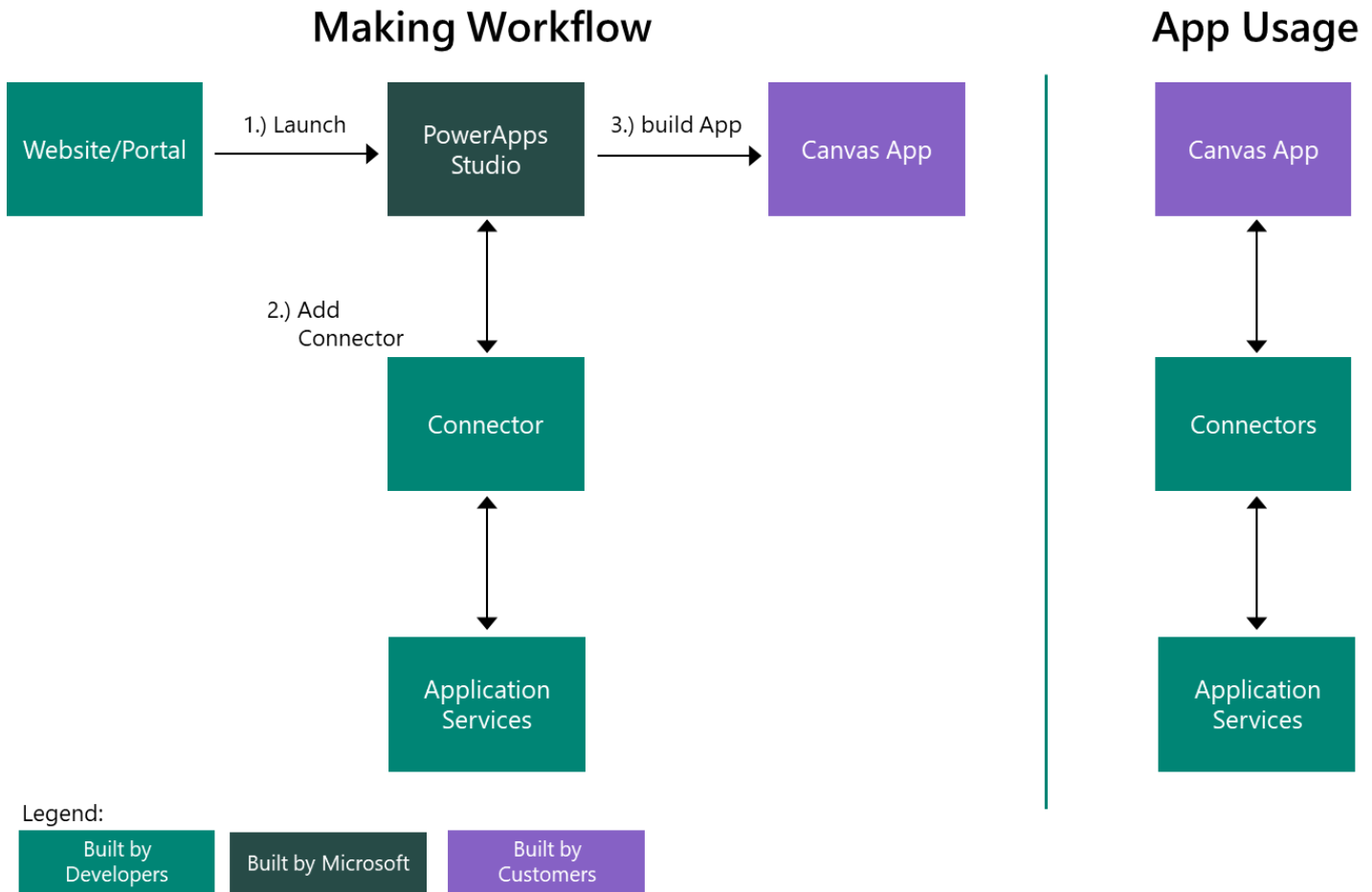
Canonical Scenarios

As you look to enable your customers to become makers, there are three different types of scenarios (which can be pursued simultaneously).

Enabling makers to build standalone apps on your services

The basic scenario is launching your users into PowerApps Studio from your own experiences. While they can build standalone apps by navigating to PowerApps Studio themselves, launching the experience directly from your experience improves discoverability and significantly lowers the barrier to entry, making it much more likely for your customers to realize they can access your core services and build standalone applications deepening their engagement with your services.

The following illustration outlines the different components involved in the solution and who builds them:



Components

- **Built by developers**
 - **Maker Portal** – This is where your customers discover and launch into PowerApps Studio. This experience is typically located in a centralized administration portal or from a settings page within your applications.
 - **PowerApps Connector** – This component enables makers to consume your core services. It acts as the adapter to translate between your service and the interfaces required by PowerApps Studio and the PowerApps runtime.
 - **Application Services** – These are the core services that are exposed within your application (i.e., data, business logic, etc.). By enabling makers to access this data and business logic, they can create experiences that seamlessly synchronize with your applications.
- **Other Components**
 - **PowerApps Studio** – This is the premiere low-code or no-code authoring experience that your customers can use to create apps.
 - **PowerApps** – These are the applications that you newly minted makers create.

Making Workflow

- 1) **Launch PowerApps Studio** – Inside of the application or portal, you can provide an affordance (e.g., a button) for your customers to launch PowerApps Studio. When they click this button, the studio launches into either a new blank canvas app or to an existing canvas app.
- 2) **Add Custom Connector** – Once your customer is in PowerApps Studio, the first thing they need to do is add your custom connector so that they can access your services.
- 3) **Build App** – Now that your customer can get data from your services, they can proceed to layout the user interface and building their app. Once done, they click save and publish the app so that other users in their organization can use it.

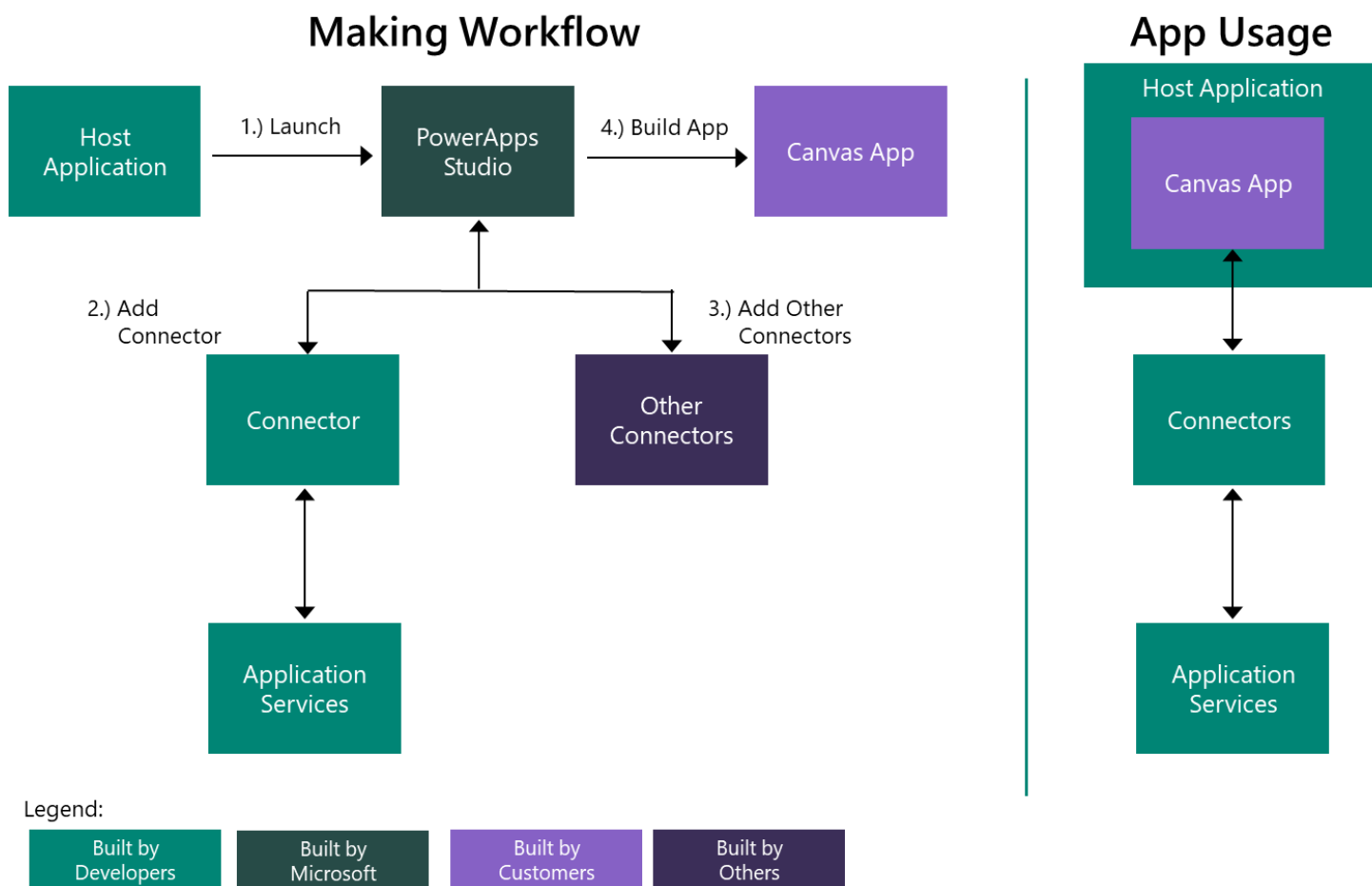
App Usage

Once the app gets created, it acts as a standalone application. Your customer can interact with it directly, which in turn use the connector to send and receive data and to apply the logic that is in your services.

Enabling makers to customize and extend your app

It has similarities with the previous concept in that your customers have the same direct navigation to PowerApps Studio where they continue to engage with your services. The difference in this scenario is that your customers' focus is on augmenting your experience by bringing in additional data and logic into your application's experience. This means that your customers can accomplish more of their work without leaving your application making it indispensably integrated into their workflow.

The following illustration outlines the different components involved in the solution and who builds them:



Making Workflow:

Much of the workflow is the same, so the below only calls out noteworthy changes.

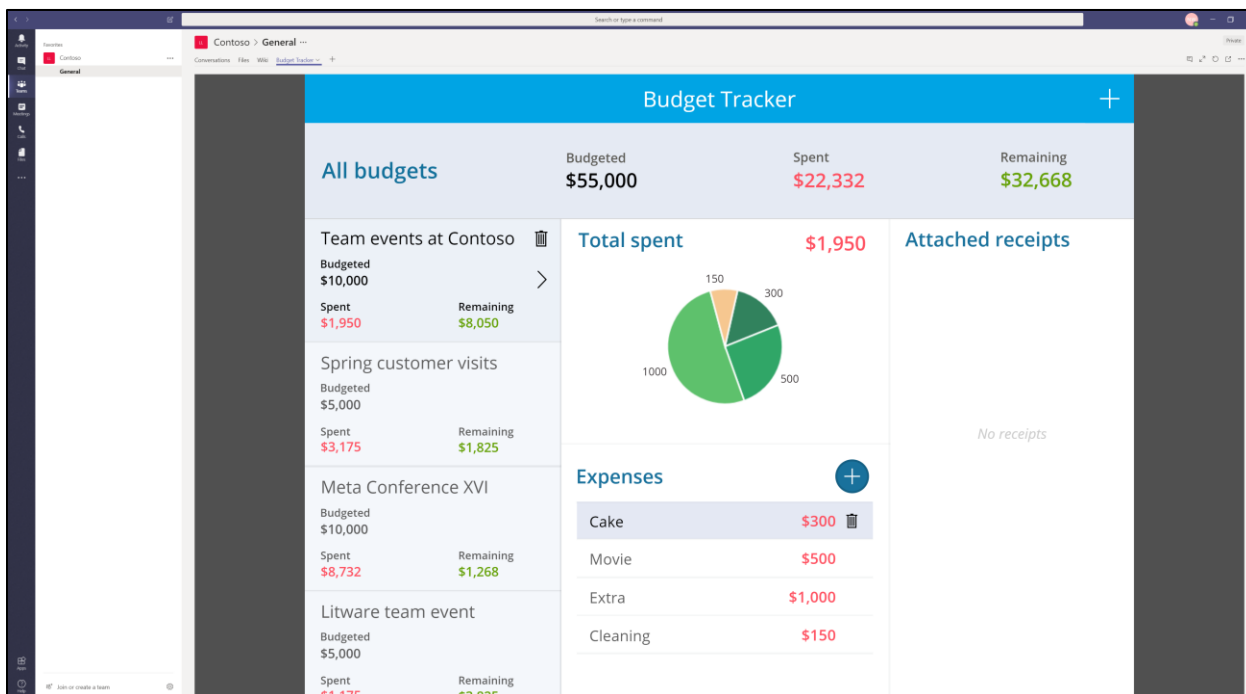
- 1) **Launch PowerApps Studio** –The key difference is that the Host Application needs to record the application identifier that is returned from the studio when the user saves/publishes the application. This identifier is later used to select which application(s) should be embedded in the host app during normal application usage.
- 2) **Add Custom Connector** – No change.
- 3) **Add Other Connectors** – As your customers extend your app, they can use the 230+ connectors to bring additional data and services into the experiences within your application.
- 4) **Build App** – No change.

App Usage

The canvas app has now been designed to work solely with your host application. There are three design patterns that are common how you seamlessly light-up the extensions your customers have made.

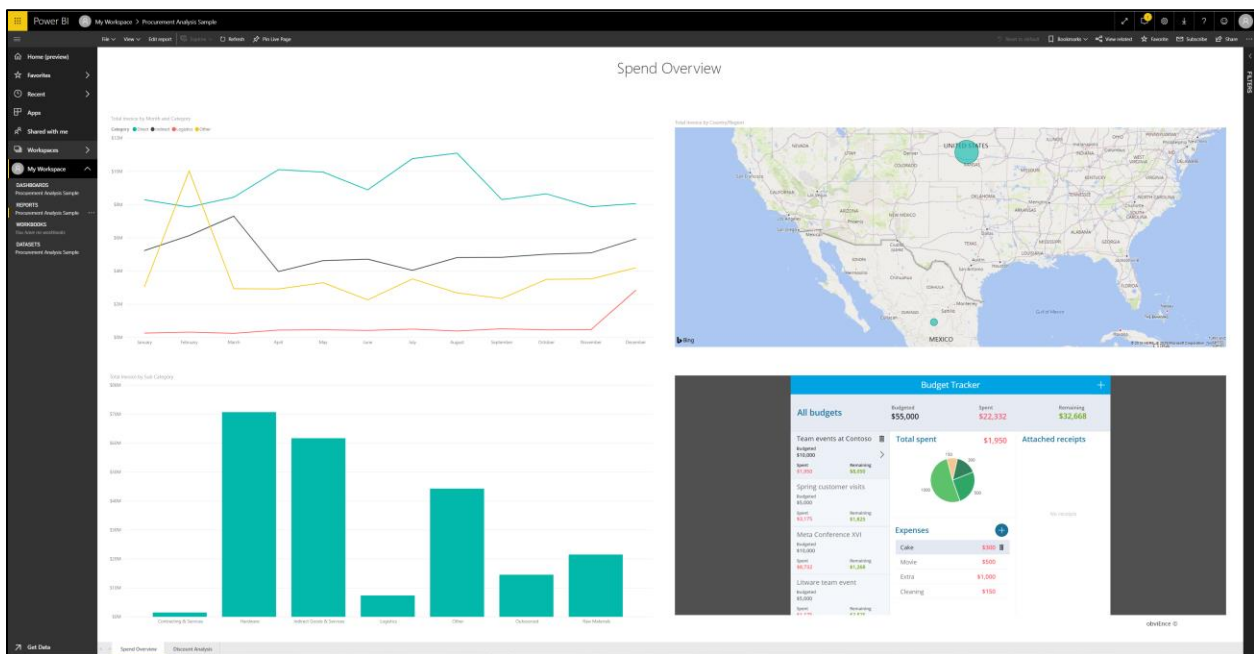
Additional Tab/Pane of UI

An example of this is the **Teams** UI where end-users optionally add an extra tab to their team channel.



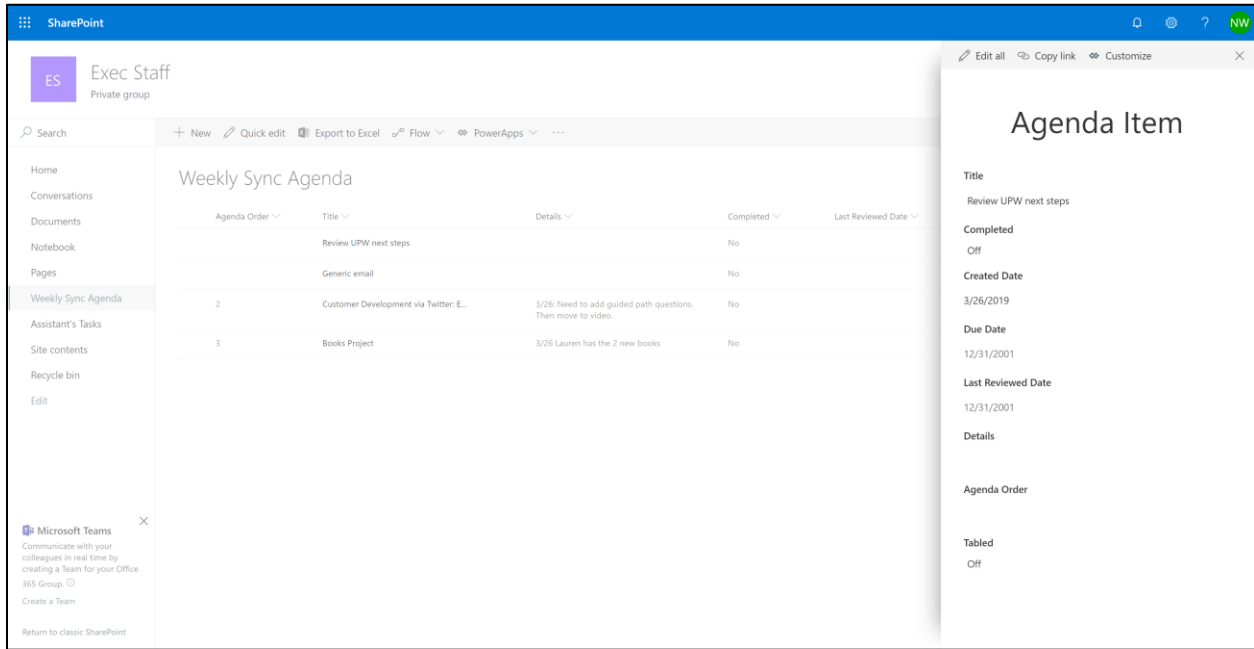
Dashboard / Customer Controlled Layout

An example of this is the **PowerBI** UI where end-users draw a box within a report that defines the region where the canvas app should be displayed:



Transient UI

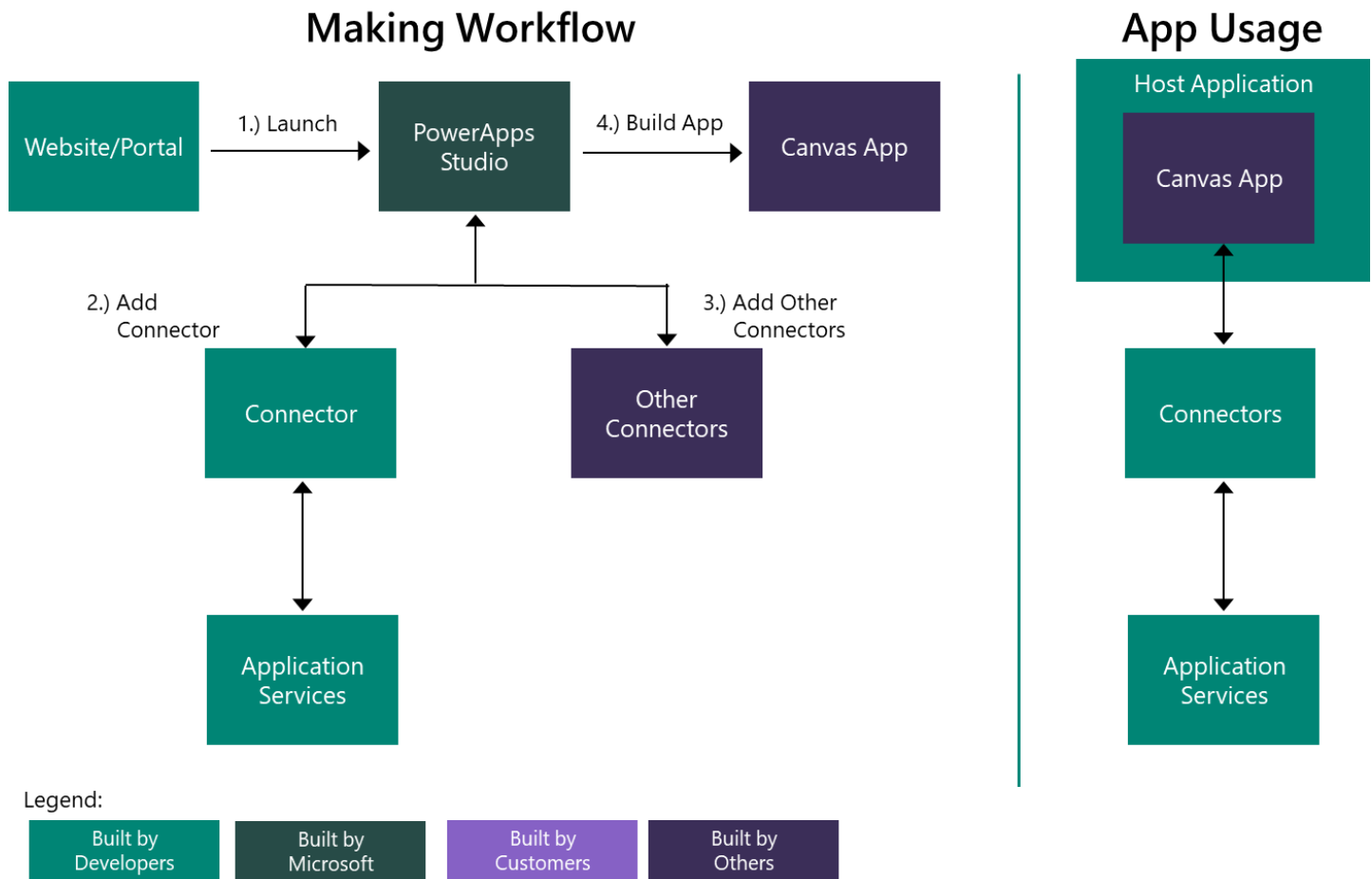
SharePoint uses transient UI to display the custom forms that the users have created for SharePoint Lists. In this case, when the user clicks on an item, the custom form is presented in a flyout that slides on the screen.



Enabling a maker extension ecosystem for your application

This scenario is all about enabling a new wider set of people to build solutions based on the pattern of problems your customers face. With this expanded audience closing the gaps within specific niches, you can employ the 80/20 rule. You can keep your focus on the critical scenarios that require your focus and cut across your customers while resting assured your ecosystem of makers is helping your application work seamlessly nearly out of the box in your customers workflows.

The following illustration outlines the different components involved in the solution and who builds them:



Making Workflow

This workflow is identical to the customization or extension one above. The only real difference is that the starting point is likely a centralized portal rather than the application itself.

App Usage

While the day to day usage of the canvas app as an extension is identical to the one above, there is a difference in the acquisition and provision steps. There needs to be a mechanism (either direct or facilitated) for your customer to acquire the application and register it with your Host App. It is done either in the Admin Portal or a settings pane within your application. Once the registration is complete, your application can light up the experience in the same way as above.

Hosting and Licensing Considerations

There are two different hosting models on how you can enable the above scenarios, each with their own licensing considerations impacting your customers in terms of which credentials they use to authenticate and use with PowerApps. Said differently, your choice of model dictates whether your customers will login to PowerApps Studio and into the hosted canvas apps with the same credentials as the ones they use for **Office 365** and **Dynamics 365** or with credentials that have provisioned within your tenant.

PowerApps hosted in your customers tenant

Your customers create accounts within their tenant and purchase licenses from Microsoft. When they authenticate with PowerApps Studio, they automatically begin building experiences hosted in their tenant. If your application is either a multi-tenant **SaaS** application or deployed within your customers' tenant, your customers can use the same credentials to log into your application and PowerApps Studio as they do to log into **Office 365** and **Dynamics 365**. Additionally, your customers have all the benefits of having a full license (i.e., they can make use of custom apps for other workloads).

PowerApps hosted in your tenant

In this model, you create accounts in your tenant and purchase licenses on your customers' behalf. Your customers use the credentials you provide to login to PowerApps Studio which is different than the credentials they use for logging into **Office 365** or **Dynamics 365**. This option is well suited for products that are targeting customers who don't want to manage the environments themselves.

Note: This model could also be employed by those in your maker extension ecosystem.

Quick Start Scenarios

Launching PowerApps Studio

Get the PowerAppsAuthoringSdk NPM package using the command `npm install @microsoft/powerappsauthoringsdk`

Use the following code to launch the MakerSession

```
// For Typescript:
// import * as PowerAppsAuthoringSdk from "@microsoft/powerappsauthoringsdk";
// For Javascript:
// PowerAppsAuthoringSdk is a global variable.
// Define the source parameter used for helping respond to support tickets

let options = {
  hostName: "AuthoringSdkDocumentation"
}

// Step 1: Initialize the SDK

PowerAppsAuthoringSdk.initAsync(options).then(function() {
  // Define a variable to represent the template that will be used when creating a new app
  (optional)

var template = new
PowerAppsAuthoringSdk.StandardTemplate(PowerAppsAuthoringSdk.FlexibleLayouts.VariableHeight);

// Step 2: Define the options for launching the MakerSession

var makerOptions: CreateAppOptions = {
  formFactor: PowerAppsAuthoringSdk.FormFactor.Phone,
  appTemplate: template,
  environmentId: "*****",
  solutionId: "*****",
  // Visit https://github.com/Microsoft/CorrelationVector-JavaScript.
  // for more information on how to generate the correlation vector.
  externalCorrelationId: "qPsdftu.1"
};
```

```
// Step 3: Start the actual session
```

```
PowerAppsAuthoringSdk.MakerSession.createAppAsync(makerOptions).then(function(makerSession) {
```

```
    // Step 4: Subscribe to lifecycle events and simply show the appId for the app
    makerSession.appSaved.subscribe(function(appInfo) {
        alert("The user saved app with Id: " + appInfo.appId);
    });
    makerSession.appPublished.subscribe(function(appInfo) {
        alert("The user published app with Id: " + appInfo.appId);
    });
}).catch(function(error) {
    alert("Failed");
});
}).catch(function(error) {
    alert("InitAsync Failed: " + error.toString());
});
```

Embedding a canvas app

Get the PowerAppsAuthoringSdk NPM package using the command `npm install @microsoft/powerappsauthoringsdk`

Use the following code to create a Player instance

```
// For Typescript:
// import * as PowerAppsPlayerSdk from "@microsoft/powerappsplayersdk";
// For Javascript PowerAppsPlayerSdk is a global variable.
// Set the host name
let initializerOptions: SdkInitializer = {
    hostName: "PlayerSDKDocumentation"
}
//Step 1: Initialize SDK
initAsync(initializerOptions).then(() => {
    let options: PlayerOptions = {
        parentContainerId: "hostIframe",
        externalCorrelationId: "correlationId"
    }

    // Step 2: Initialize the app by AppId or LogicalName
    let player = Player.initPlayerByAppId(options, 'appid');

    // Step 3: Render the App
    player.renderApp();
}).catch((error) => {
    console.log("SDK initialization failed" + error.toString());
});
```


Canvas app embedding SDKs

PowerApps Authoring SDK

This section contains reference documentation for the PowerApps Authoring SDK which can be used via TypeScript and JavaScript. The Authoring SDK enables developers to:

1. Launch PowerApps Studio to allow the user to create or open canvas apps in the specified environment.
2. Specify the form-factor (tablet or mobile) for a new application along with predefined templates.
3. Subscribe to events from the Studio or Maker (like Studio Launched, App Saved, App Published) that describe the user's interaction with the app and provides information (i.e., the application id) about the created app.
4. Set schema for the data which can be passed to the application during authoring/runtime.
5. Set data on the application during authoring time, this data shows up during authoring time but is not saved with application.
6. Set host method definition which allows the application to callback into the host.

Developers can use the application id to reopen the app in PowerApps Studio and to embed the canvas app in their application using [PowerApps Player SDK](#)

Classes

Class	Description
MakerSession	Represents one PowerApps Studio session in which a new app is being created, or an existing app is being edited.
StandardTemplate	Provides a way to describe a layout that can be applied to the newly created app. The layout is described using the type Layout .
EventHook	The class implements the publisher/subscriber pattern so that listeners can subscribe and unsubscribe to Maker events.

Interfaces

Interface	Description
SdkInitializer	Describes the options used for initializing the SDK.
MakerSessionAppInfo	Interface representing the data returned from the appSaved and appPublished events from MakerSession .

[EditAppOptions](#) Describes the possible option that can be used to initialize a [MakerSession](#).

[CreateAppOptions](#) Allows users to specify the options to create a basic PowerApp.

[CreateAppWithSQLConnectionOptions](#) Allows users to specify the options to create a PowerApp with a SQL Connection.

Global Methods

Name	Description
initAsync	Initializes the SDK. You should call this method before using any other features from the SDK. Calling this method more than once will fail.

Enums

Enum	Description
VerticalLayouts	Represents the various vertical layouts available with the StandardTemplate .
HorizontalLayouts	Represents the various horizontal layouts available with the StandardTemplate .
FlexibleLayouts	Represents the various flexible layouts available with the StandardTemplate .
FormFactor	Represents the available form factor targets for a new application.

Types

Type	Description
Layout	Union type representing VerticalLayouts , HorizontalLayouts , FlexibleLayouts . It is used to define the parameter requirement on the StandardTemplate constructor.

[Template](#)

This type is defined for forward compatibility reasons and will become a Union Type as if/when additional template types are supported.

Global Methods

initAsync

Initialization method for the SDK which should be called before using any other features from the SDK. Calling this method multiple times will fail.

Syntax

```
export async function initAsync(options: SdkInitializer): Promise<void>
```

Parameters

Name	Type	Required	Description
sdkInitializer	SdkInitializer	Yes	Describes the options used for initializing the SDK.

Return Value

Returns a promise which is resolved on successful initialization and rejected on failure.

Examples

Initialize the SDK

```
// For Typescript:
// import * as PowerAppsAuthoringSdk from "@microsoft/powerappsauthoringsdk";
// For Javascript PowerAppsAuthoringSdk is a global variable.

const sdkInitializer: PowerAppsAuthoringSdk.SdkInitializer = {
  hostName: "DemoApp"
}
await PowerAppsAuthoringSdk.initAsync(sdkInitializer);
```

Classes

MakerSession

Represents a PowerApps Studio session in which a new app is being created, or an existing app is being edited.

Syntax

```
export class MakerSession {  
    public static editAppAsync(editAppOptions: EditAppOptions): Promise<MakerSession>  
    public static createAppAsync(createOptions: CreateAppOptions): Promise<MakerSession>  
    public static createAppWithSQLConnectionAsync(createOptions: CreateAppWithSQLConnectionOptions)  
    public async setSchemaAsync(schema: PowerAppsSchema): Promise<void>  
    public async setDataAsync(data: PowerAppsData): Promise<void>  
    public async setHostMethodDefinition(hostMethodDef: string): Promise<void>  
    public disposeAsync(): Promise<void>;  
  
    public appSaved: Core.Embedding.EventHooks<MakerSessionAppInfo>;  
    public appPublished: Core.Embedding.EventHooks<MakerSessionAppInfo>;  
}
```

Methods

Method	Description
editAppAsync	Starts a new maker session in edit mode.
createAppAsync	Starts a new maker session to create a new blank app.
createAppWithSQLConnectionAsync	Start a new maker session to create a PowerApp with data from a Sql Connection.
setSchemaAsync	Sets the schema for the data which can be passed to the PowerApp during runtime using setData method or using setDataAsync method during authoring time.
setDataAsync	Sets the data on the PowerApp during authoring time.
setHostMethodDefinition	Allows the user to pass in a WADL file describing the callback methods (on host) which the PowerApp can call.
disposeAsync	Terminates or disposes of an ongoing maker session. Since at a time only a single maker session is supported, to create a second maker session the first session must be disposed of using this method. Use this method with caution as the user may lose unsaved changes in the authoring session.

Event

Method	Description
appSaved	Event is triggered when the user clicks the save button in the PowerApps Studio.
appPublished	Event is raised when the user clicks the publish button in PowerApps Studio. Note, the first time an app is saved, both the appSaved and appPublished events are fired without the user having to click on Publish App button explicitly.

EditAppAsync

Starts a new maker session, in edit mode.

Syntax

```
public static editAppAsync(editAppOptions: EditAppOptions):  
    Promise<MakerSession>;
```

Parameters

Name	Type	Required	Description
editAppOptions	EditAppOptions	Yes	Describes the possible option that can be used to initialize a <i>MakerSession</i> in edit mode.

Return Value

Returns a promise which resolves to [MakerSession](#) instance. If there are any failures while starting a new session, this promise gets rejected.

CreateAppAsync

Starts a new maker session, and creates a new blank PowerApp.

Syntax

```
public static createAppAsync(createOptions: CreateAppOptions): Promise<MakerSession>
```

Parameters

Name	Type	Required	Description
------	------	----------	-------------

createOptions	CreateAppOptions	Yes	Describes the possible option that can be used to create a new blank PowerApp.
---------------	----------------------------------	-----	--

Return Value

Returns a promise which resolves to [MakerSession](#) instance. If there are any failures while starting a new session, this promise gets rejected.

CreateAppWithSQLConnectionAsync

Starts a new maker session, and creates a new PowerApp using the data from the SQL Connection.

Syntax

```
public static createAppWithSQLConnectionAsync(createOptions: CreateAppWithSQLConnectionOptions)
: Promise<MakerSession>
```

Parameters

Name	Type	Required	Description
createOptions	CreateAppWithSQLConnectionOptions	Yes	Describes the possible option that can be used to create a new PowerApp using the data from the SQL Connection.

Return Value

Returns a promise which resolves to [MakerSession](#) instance. If there are any failures while starting a new session, this promise gets rejected.

SetSchemaAsync

Sets the schema for the data that can be passed to the PowerApp during Authoring using setDataAsync method or during runtime using the setData method.

Syntax

```
public async setSchemaAsync(schema: PowerAppsSchema): Promise<void>
```

Parameters

Name	Type	Required	Description
schema	PowerAppsSchema	Yes	Describes the data schema.

Return Value

Returns a promise which is resolved on successfully setting the data schema on the PowerApp, rejected otherwise.

SetDataAsync

Sets the data on the PowerApp during Authoring.

This data is not saved, it only serves the purpose of allowing the user to visualize the data.

It is the user's responsibility to ensure that the data adheres to the schema which was previously set.

Syntax

```
public async setDataAsync(data: PowerAppsData): Promise<void>
```

Parameters

Name	Type	Required	Description
data	PowerAppsData	Yes	Sets the data on the PowerApp.

Return Value

Returns a promise which is resolved on successfully setting the data on the PowerApp, rejected otherwise.

SetHostMethodDefinitionAsync

Sets the host method definition described as a WADL file on the PowerApp.

PowerApp can call into this method.

Syntax

```
public async setHostMethodDefinitionAsync(hostMethodDef: string): Promise<void>
```

Parameters

Name	Type	Required	Description
hostMethodDef	string	Yes	Sets the host method definition on the PowerApp.

Return Value

Returns a promise which is resolved on successfully setting the data on the PowerApp, rejected otherwise.

disposeAsync

Terminates or disposes of an ongoing maker session. Since at a time only a single maker session is supported, to create a second maker session the first session must be disposed of using this method. On successful completion of this method, all event listeners get automatically unsubscribed. Use this method with caution as the user may lose **unsaved** changes in the authoring session. This method takes no argument.

Syntax

```
public disposeAsync(): Promise<void>;
```

Parameters

This method takes no parameters.

Return Value

Returns a promise which is resolved when the current maker session is disposed of. Once resolved, the user is free to call `startAsync` to create a new session. A promise gets rejected if there were errors in the disposing of the session. The user can try to dispose of the session again.

Examples

```
// Assuming that the application has been saved and published in the Maker Studio and
// now the user wishes to start a new Maker Session to create/edit another application.

// Note that calling startAsync while a session is still active results in an error.
// so the below code given the above assumption that a session exists will throw an error:
// await PowerAppsAuthroingSdk.MakerSession.startAsync(makerSessionOptions); // error

await makerSession.disposeAsync();

// After this the user is allowed to call MakerSession.startAsync method again.
// i.e. once disposeAsync succeeds the below code will succeed:
// await PowerAppsAuthroingSdk.MakerSession.startAsync(makerSessionOptions); // OK
```

appSaved

Event is raised when the user clicks the save button in PowerApps Studio. The *appSaved* event is of type *EventHook<MakerSessionAppInfo>*. Which means when the event is raised, it has a single parameter which conforms to the *MakerSessionAppInfo* interface which provides the *appId*.

This event callback can be used to get Session state, which enables warning the user to publish the application before disposing of the current session if the user has not yet published the app before trying to dispose of the session.

Syntax

```
public appSaved: Core.Embedding.EventHooks<MakerSessionAppInfo>;
```


Example

Subscribe/Unsubscribe from MakerSession Events

```
let makerSession = await authoringSdk.MakerSession.startAsync(makerSessionOptions);

// Subscribe to application saved event
function appSavedCallback(makerSessionAppInfo) {
    Console.log("User saved application with id: " + makerSessionAppInfo.appId);
}

makerSession.appSaved.subscribe(appSavedCallback);

// When the caller no longer wishes to receive callbacks on appSaved events, he can
// Unsubscribe from application saved event
makerSession.appSaved.unsubscribe(appSavedCallback);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
makerSession.appSaved.unsubscribeAll();
```

appPublished

Event is raised when the user clicks the publish button in PowerApps Studio. The *appPublished* event is of type *EventHook<MakerSessionAppInfo>*. Which means when the event is raised, it has a single parameter which conforms to the *MakerSessionAppInfo* interface which provides the *appId*.

The primary reason to subscribe to the event is to save the *appId* so that you can use it to identify the app and [integrate it into your website and other services](#).

Syntax

```
public appPublished: Core.Embedding.EventHooks< MakerSessionAppInfo>;
```

Example

Subscribe/Unsubscribe from MakerSession Events

```
let makerSession = await authoringSdk.MakerSession.startAsync(makerSessionOptions);

// Subscribe to application published event
function appPublishedCallback(makerSessionAppInfo) {
    Console.log("Enable the extension with id: " + makerSessionAppInfo.appId);

    // You can now use makerSessionAppInfo.appId to embed the app in an iFrame and
    // run the app for the user. For more information visit:
    // https://docs.microsoft.com/en-us/powerapps/maker/canvas-apps/embed-apps-dev.
}

makerSession.appPublished.subscribe(appPublishedCallback);
```

```
// // When the caller no longer wishes to receive callbacks on appPublished event, he can
// unsubscribe from the event.
makerSession.appPublished.unsubscribe(appPublishedCallback);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
makerSession.appPublished.unsubscribeAll();
```

StandardTemplate

Provides a way to describe a layout that can be applied to the newly created app.

Syntax

```
export class StandardTemplate {
    public constructor(layout: Layout);
    public getTemplateName(): string;
}
```

Methods

Method	Description
Constructor	Accepts a single layout parameter as input that defines the type of template used.
getTemplateName	Returns the string representation of the layout.

Constructor

Syntax

```
public constructor(layout: Layout);
```

Parameters

Name	Type	Required	Description
layout	Layout	Yes	Describes the type of layout to create.

Return Value

A new *StandardTemplate* object.

Examples

Creating a Vertical Layout

```
const template = new
PowerAppsAuthoringSdk.StandardTemplate(VerticalLayouts.Vertical_OneText);

const makerSessionOptions = {
  appTemplate: template
}

authoringSdk.MakerSession.startAsync(makerSessionOptions).then((session) => {
  makerSession = session;
});
```

getTemplateName

Returns the string representation of the layout.

Information purpose only.

Syntax

```
public getTemplateName(): string;
```

Return Value

Returns the string representing the template layout.

Examples

```
const template = new PowerAppsAuthoringSdk.StandardTemplate(VerticalLayouts.Vertical_OneText);
console.log(template.getTemplateName()); // "BrowseLayout_Vertical_OneTextVariant_ver4.0"
```

EventHook

The class implements the publisher or subscriber pattern so that listeners can subscribe to events using the subscribe method and unsubscribe using the unsubscribe method.

Methods

Method	Description
subscribe	Registers a function that conforms to the <i>EventListener</i> type that will be called when the event is raised.
unsubscribe	Registers a function that conforms to the <i>EventListener</i> type that will be called when the event is raised.
unsubscribeAll	If there are one or more listeners subscribed for an event, using <i>unsubscribeAll</i> will unsubscribe all the listeners.

subscribe / unsubscribe / unsubscribeAll

Syntax

```
eventHook.subscribe(listener);  
eventHook.unsubscribe(listener);  
eventHook.unsubscribeAll();
```

Parameters

Name	Type	Required	Description
Listener	EventListener<T>	Yes	A function that will be called when the event is raised.

Return Value

None

Example

Subscribe/Unsubscribe from MakerSession Events

```
let makerSession = await authoringSdk.MakerSession.startAsync(makerSessionOptions);  
  
// Subscribe to application published event  
function appPublishedCallback(makerSessionAppInfo) {  
    Console.log("Enable the extension with id: " + makerSessionAppInfo.appId);  
}  
makerSession.appPublished.subscribe(appPublishedCallback);  
  
// Unsubscribe from application published event  
makerSession.appPublished.unsubscribe(appPublishedCallback);  
// If there are multiple listeners  
makerSession.appPublished.unsubscribeAll();
```

Interfaces

SdkInitializer

This interface defines the contract for initializing the SDK, which must be done before using any other functionality from the SDK.

Syntax

```
export interface SdkInitializer {  
    hostName: string;  
}
```

Properties

Name	Type	Required	Description
hostname	string	Yes	This parameter is used primarily to help identify your host application when responding to support tickets.

MakerSessionAppInfo

This interface defines the set of properties that are available on objects returned by events (e.g., appSaves, appPublished) that describe an app. The *appId* described below can be used to relaunch PowerApps Studio to continue editing the app or to embed the app using [PowerAppsPlayerSdk](#).

Properties

Property	Description
appId	The application id of the application being edited or created in the Maker session.
environmentId	The environment in which the application was last saved/published.
logicalName	The logical name for the application which is unique and consistent across tenant.
tenantId	The tenant id in which the application was last saved/published.

EditAppOptions

This interface defines the contract for the options that can be used to initialize a Maker session for editing an app.

Properties

Name	Type	Required	Description
appId	string	Yes	The application id which the user wants to edit.

CreateAppOptions

This interface defines the contract for the options that can be used to initialize a Maker session for editing an app.

Properties

Name	Type	Required	Description
------	------	----------	-------------

appTemplate	Template	No	Defines the specific template/layout that should be used for the initial screen your users will see when PowerApps Studio is used to create a new app. If unspecified this will default to no template.
displayName	String	No	The application display name. If the application display name is not valid/unique a system generated name will be used. The user can change this name when saving the app.
environmentId	string	No	This property specifies the environment in which the new application will be created. Note if a <i>solutionId</i> is specified, this property is ignored.
externalCorrelationId		No	This parameter is optional, but it is recommended as it helps to troubleshoot and enables better support if issues should arise.
formfactor	FormFactor	No	This property defines the target form factor that a new application should target, primarily affects the layout of the app. If unspecified this will default to Phone layout.
solutionId	string	No	When a new app is to be created, this property specifies the solution that the app should be created in. Note, if a <i>solutionId</i> is specified, the <i>environmentId</i> is ignored. If no <i>solutionId</i> is specified and a new app is being created, it will be created outside of a PowerApps solution . PowerApps solutions greatly enhance our current app lifecycle story.

CreateAppWithSQLConnectionOptions

This interface defines the contract for the options that can be used to initialize a Maker session for editing an app.

Properties

Name	Type	Required	Description
appTemplate	Template	No	Defines the specific template/layout that should be used for the initial screen your users will see when PowerApps Studio is used to create a new app. If unspecified this will default to no template.
connectionName	String	Yes	The connection name of the SQL Connection. This is in the form of a GUID.
databaseName	String	Yes	The database name to which the connection connects.
displayName	String	Yes	The application display name. If the application display name is not valid/unique a system generated name will be used. The user can change this name when saving the app.
environmentId	string	Yes	This is the environment in which the connection is created and will be the environment in which the application will be created.
externalCorrelationId		No	This parameter is optional, but it is recommended as it helps to troubleshoot and enables better support if issues should arise.
formfactor	FormFactor	No	This property defines the target form factor that a new application should target, primarily affects the layout of the app. If unspecified this will default to Phone layout.
serverName	String	Yes	This is the server name to which the connection connects.
solutionId	string	No	When a new app is to be created, this property specifies the solution that the app should be created in. Note, if a <i>solutionId</i> is specified, the <i>environmentId</i> is ignored. If no <i>solutionId</i> is specified and a new app is being created, it will be created outside of a PowerApps

[solution](#). PowerApps solutions greatly enhance our current app lifecycle story.

tableId	String	Yes	The table identifier within a particular SQL server and database.
---------	--------	-----	---

PowerAppsSchema

This defines the schema for the data which can be set during authoring using setDataAsync or during runtime using the setData call using Player SDK.

```
export type PowerAppsSchema = {  
  [columnName: string]: DataType |  
    {  
      type: DataType;  
      invariantName?: string;  
    }  
}
```

Enums

VerticalLayouts

Represents the different vertical layouts available with the *StandardTemplate*.

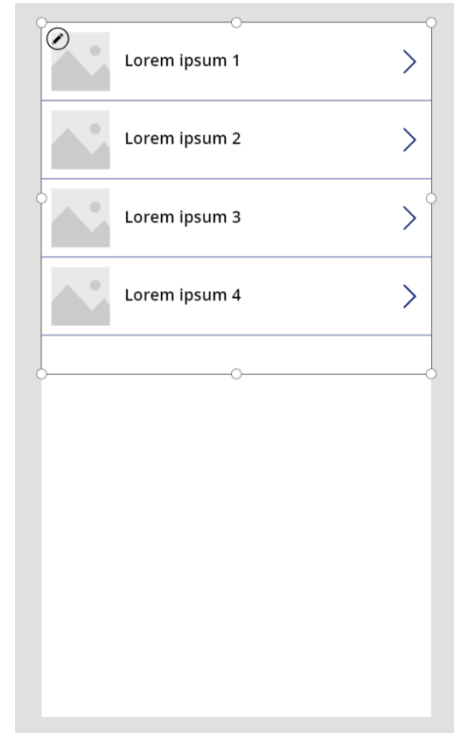
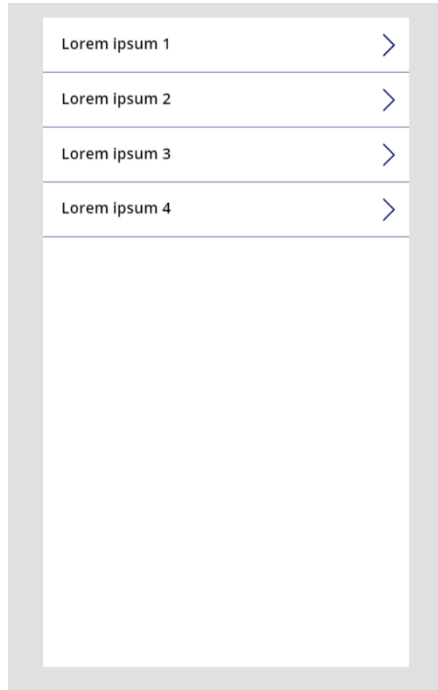
Note: Form factor and Template or layout are independent variables, any layout can be applied to Phone or Tablet. That said, HorizontalLayouts are more suited for Tablets.

The values and layout they represent are as follows:

Vertical

Vertical_OneText

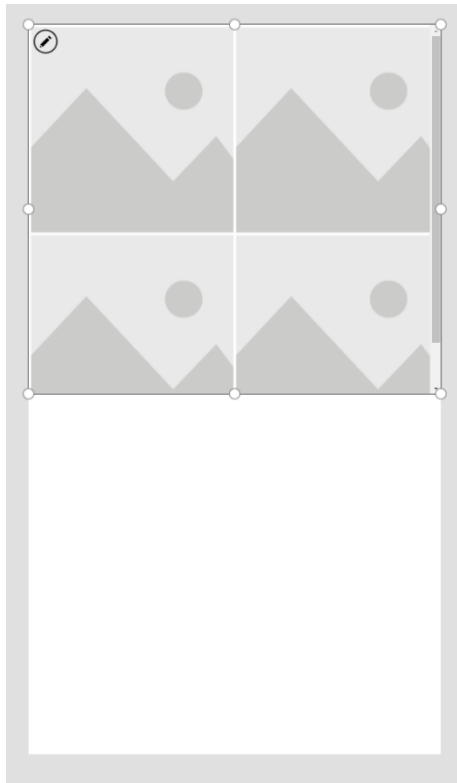
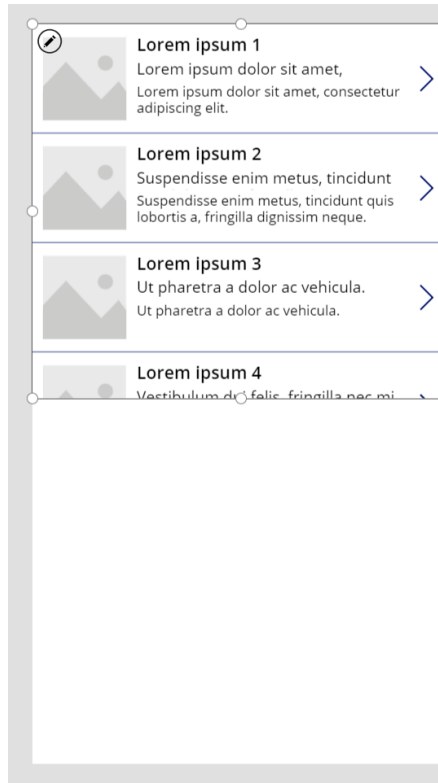
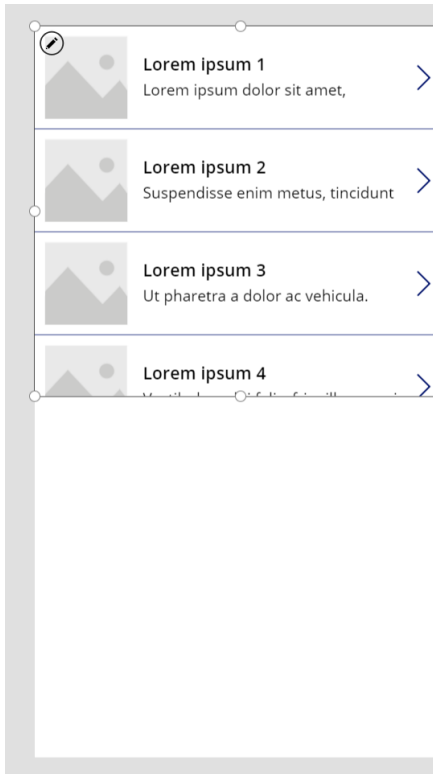
Vertical_OneText_OneImage



Vertical_TwoText_OneImage

Vertical_ThreeText_OneImage

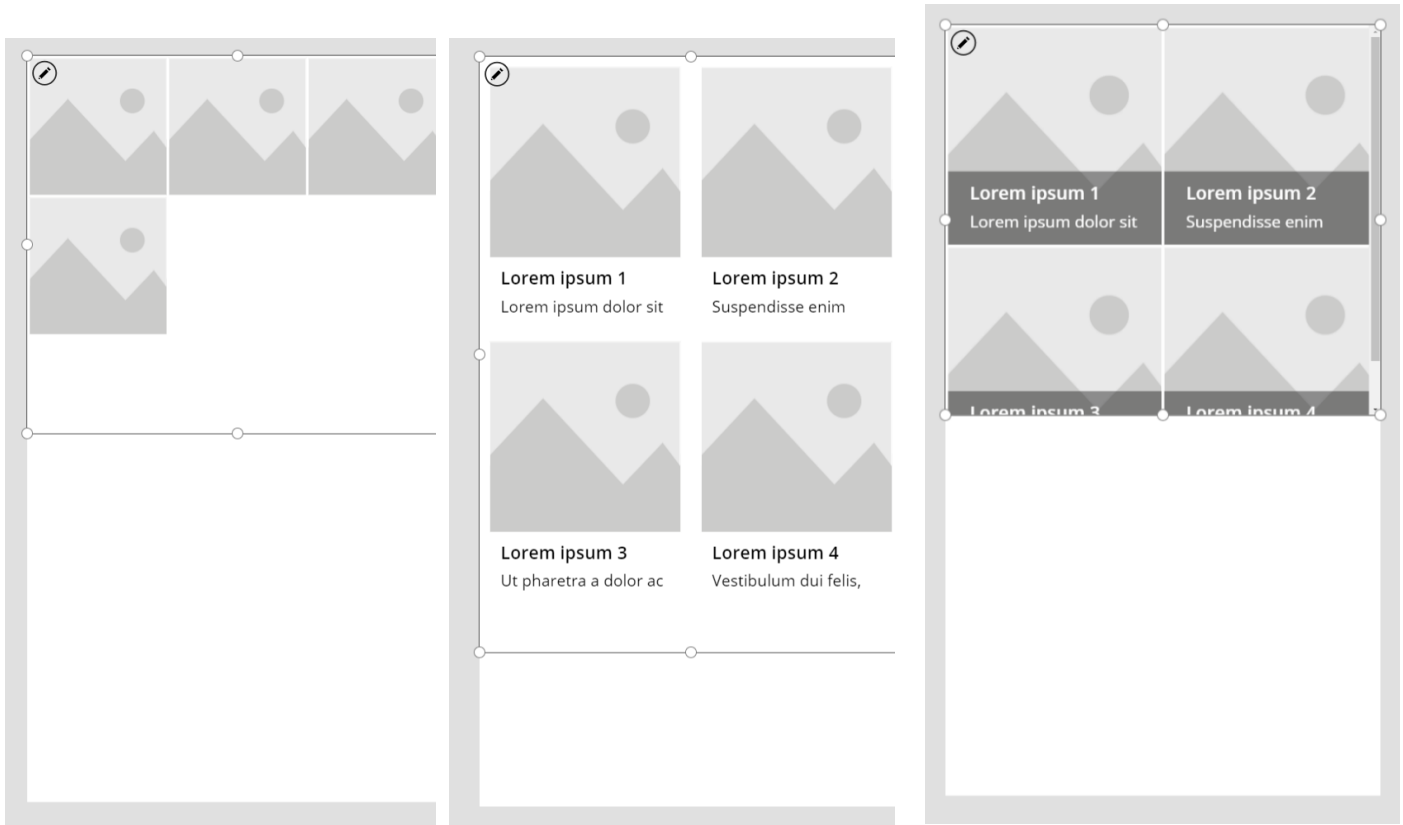
Vertical_OneImage_TwoColumn



Vertical_OneImage_ThreeColumn

Vertical_TwoText_OneImage_TwoColumn

Vertical_TwoText_OneImage_TwoColumnOverlay



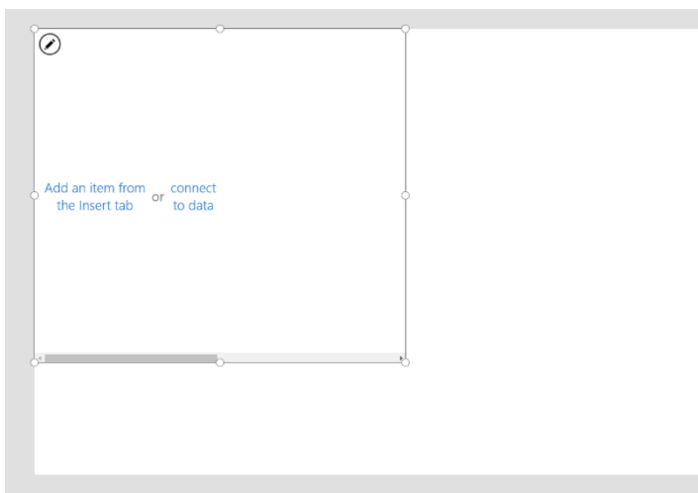
HorizontalLayouts

Represents the different horizontal layouts available with the *StandardTemplate*.

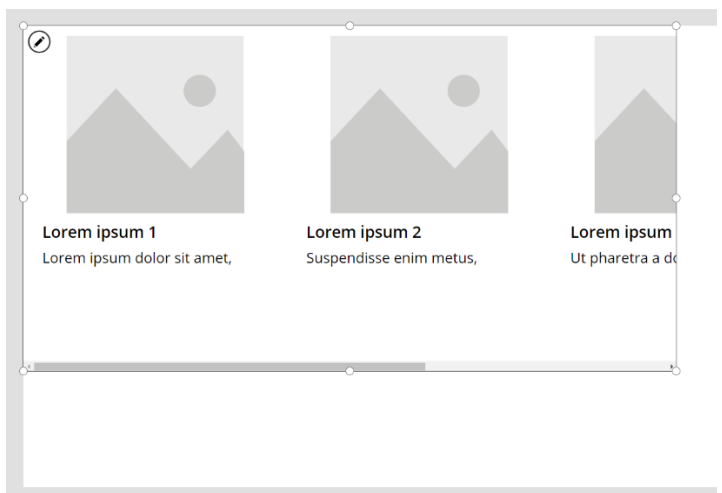
Note: Form factor and Template or layout are independent variables, any layout can be applied to Phone or Tablet. That said, *HorizontalLayouts* are more suited for tablets.

The values and layout they represent are as follows:

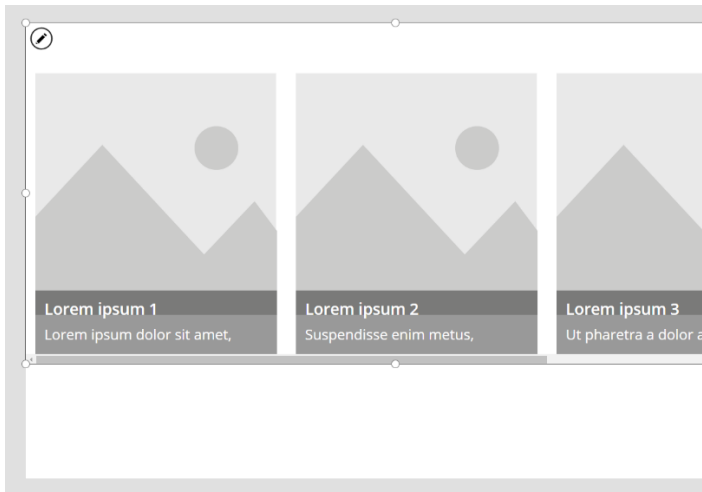
Horizontal



Horizontal_TwoText_OneImage



Horizontal_TwoText_OneImageOverlay



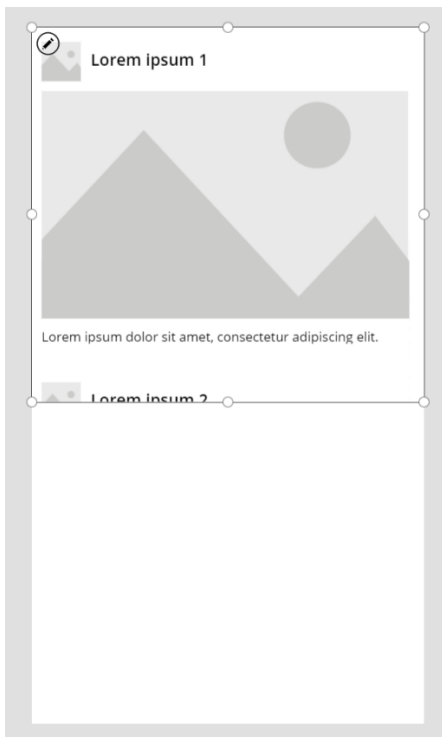
FlexibleLayouts

Represents the different flexible layouts available with the *StandardTemplate*.

Note: Form factor and Template or layout are independent variables, any layout can be applied to Phone or Tablet. That said, *HorizontalLayouts* are more suited for tablets.

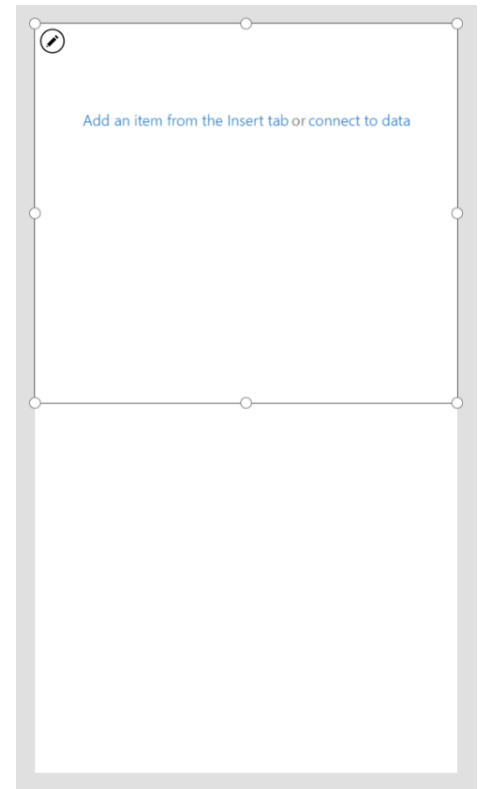
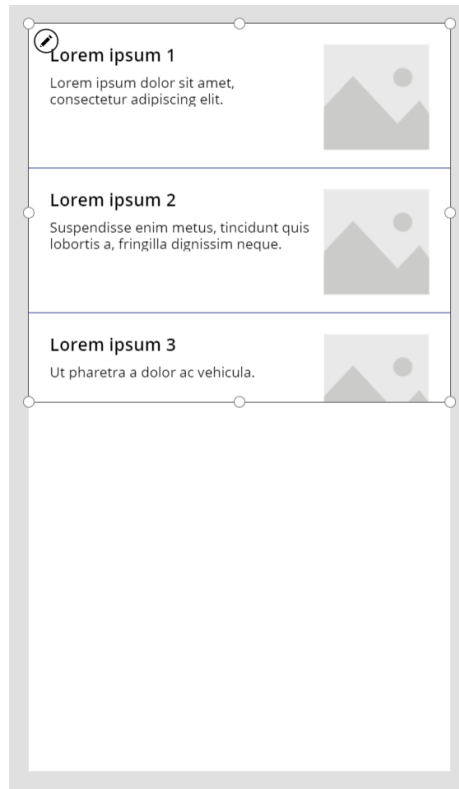
The values and layout they represent are as follows:

Flexible_SocialFeed



Flexible_NewsFeed

VariableHeight



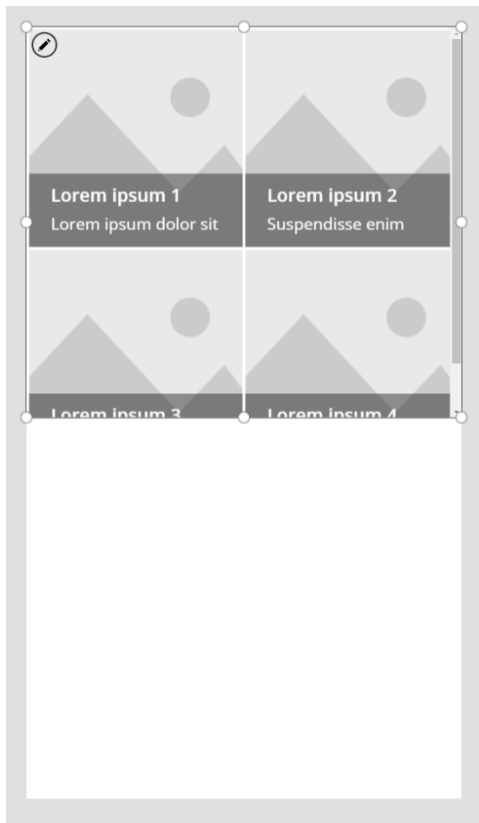
FormFactor

Represents the available form factors for the new application.

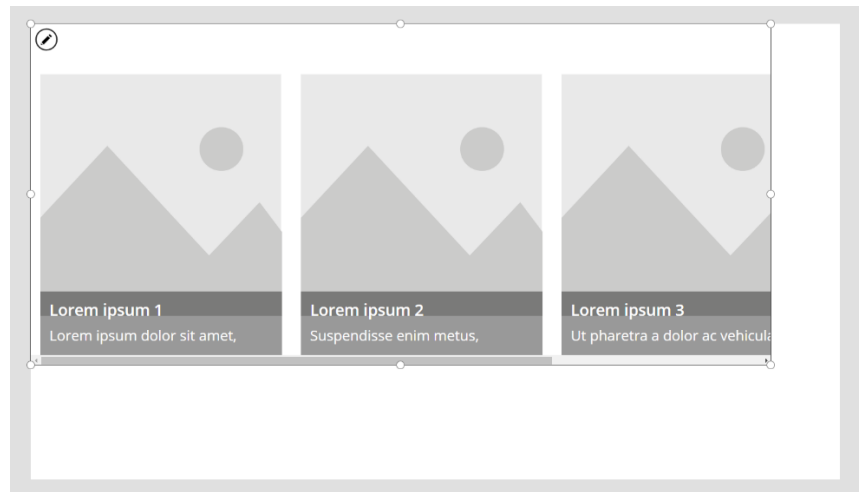
Note: Form factor and Template or layout are independent variables, any layout can be applied to Phone or Tablet. That said, *HorizontalLayouts* are more suited for tablets.

The values and layout they represent are as follows:

Phone



Tablet



Types

EventListener

Basic type representing an event listener that can receive a single object of type T.

Definition

```
export type EventListener<T> = (data?: T) => void;
```

Template

This type is defined for forward compatibility reasons and will become a Union Type if and when additional template types are supported.

Definition

```
export type Template = StandardTemplate;
```

Layout

Union type representing *VerticalLayouts*, *HorizontalLayouts*, *FlexibleLayouts*. It is used to define the parameter requirement on the *StandardTemplate* constructor.

Definition

```
export type Layout = VerticalLayouts | HorizontalLayouts | FlexibleLayouts;
```

PowerApps Player SDK

This section contains reference documentation for the PowerApps Player SDK which can be used via TypeScript and JavaScript. The Player SDK enables developers to:

1. Embed an app by App Id
2. Embed an app by Logical name, Environment Id and Tenant Id.
3. Set data on the app
4. Register host methods that can be called in the App. Schema for the methods and data can be set using [PowerApps Authoring SDK](#).
5. Subscribe to events from Player (like App created, App loaded and App Errored)

Classes

Class	Description
Player	Represents PowerApps Player instance which can be used to play an App by App Id or Logical Name.
EventHook	The class implements the publisher/subscriber pattern so that listeners can subscribe and unsubscribe to Player events.

Interfaces

Interface	Description
SdkInitializer	Describes the options used for initializing the SDK.
RGBA	Describes the RGB color with optional value for alpha. This defines a type for splash screen color.

[PlayerOptions](#) Describes the possible options that can be used to play an App.

[PlayerInfo](#) Interface represents the data returned from the [onAppLoaded](#), [onAppUnload](#) and [onAppError](#) events from Player instance

Global Methods

Name	Description
initAsync	Initializes the SDK. You should call this method before using any other features from the SDK. Calling this method more than once will fail.
isSdkInitialized	This method returns a true if the SDK is initialized and can be used to check the SDK state.

Enums

Enum	Description
SdkState	Represents the SDK states.

Types

Type	Description
AccessTokenProvider	A type that takes in an audience, a tokenCallback with the access token for that audience, an errorCallback if the access token cannot be retrieved
RecordData	A type that represents a record in PowerAppsData
TableData	A type that represents a table.
PowerAppsData	A type that can be used to set data on a PowerApp

Global Methods

initAsync

Initialization method for the SDK which should be called before using any other features from the SDK. Calling this method multiple times will fail.

Syntax

```
export async function initAsync(options: SdkInitializer): Promise<void>
```

Parameters

Name	Type	Required	Description
sdkInitializer	SdkInitializer	Yes	Describes the options used for initializing the SDK.

Return Value

Returns a promise which is resolved on successful initialization and rejected on failure.

Examples

Initialize the SDK

```
// For Typescript:  
// import * as PowerAppsPlayerSdk from "@microsoft/powerappsplayersdk";  
// For Javascript PowerAppsPlayerSdk is a global variable.  
  
const sdkInitializer: PowerAppsPlayerSdk.SdkInitializer = {  
  hostname: "DemoApp"  
}  
await PowerAppsPlayerSdk.initAsync(sdkInitializer);
```

isSdkInitialized

This method returns a true if the SDK is initialized and can be used to check the SDK state

Syntax

```
export function isSdkInitialized(): boolean;
```

Parameters

This method takes no parameters.

Return Value

Returns a boolean stating if the SDK is initialized or not.

Examples

Check if SDK is initialized

```
let isSDKInitialized = PowerAppsPlayerSdk.isSdkInitialized();
```

Classes

Player

Represents PowerApps Player instance which can be used to play an App by App Id or Logical Name.

Syntax

```
export class Player {
    public static initPlayerById(playerOptions: PlayerOptions,
                                appId: string): Player;

    public static initPlayerByLogicalName(playerOptions: PlayerOptions,
                                          logicalName: string,
                                          environmentId: string,
                                          tenantId: string): Player;

    public renderApp():void;
    public dispose():void;
    public registerHostFunction(name: string, func: any) :void;
    public setData(data: PowerAppsData): void;
    public onAppLoaded: EventHook<PlayerInfo>;
    public onAppUnload: EventHook<PlayerInfo>;
    public onAppError: EventHook<PlayerInfo>;
}
```

Methods

Method	Description
initPlayerByAppId	This method instantiates Player by App Id and returns a player instance. This instance can be used to render an app or to subscribe to the Player events. It throws an error if valid options are not passed.
initPlayerByLogicalName	This method instantiates Player by LogicalName, EnvironmentId and TenantId and returns a player instance. This instance can be used to render an app or to subscribe to the Player events. It throws an error if valid options are not passed.
renderApp	This method is used to play/render the PowerApp

dispose	This method disposes the player instance that was being used to play the App. It also, unsubscribes all the registered events.
setData	This method allows the host to set data on the host object that was created during the authoring time.
registerHostCallback	This method allows the host to register methods that can be called in the App. Schema for the methods and data can be set using PowerApps Authoring SDK while creating the App.

Event

Method	Description
onAppLoaded	Event is triggered when App is successfully loaded
onAppUnload	Event is triggered when the App is closed. This can happen before load is fired. For e.g. it happens when user does not agree to consent dialog
onAppError	Event is triggered when the App when does not load due to an error

initPlayerByAppId

This method instantiates Player by App Id and returns a player instance. This instance can be used to render an app or to subscribe to the Player events. It throws an error if valid options are not passed.

Syntax

```
public static initPlayerByAppId(playerOptions: PlayerOptions, appId: string): Player;
```

Parameters

Name	Type	Required	Description
playerOptions	PlayerOptions	Yes	Describes the possible option that can be used to initialize a <i>Player</i> .
appId	String	Yes	Id of the App that would be played. It should be a GUID.

How to get AppId

1. If you are using PowerApps Authoring SDK, app Id is returned in [appSaved](#) and [appPublished](#) event
2. In powerapps.com, on the Apps tab, click or tap the ellipsis(...), then Details- copy the App ID highlighted in red



Return Value

Returns a [Player](#) instance. If there are any failures while creating a new instance, an error is thrown.

Examples

Play a PowerApp by App Id

```
const options = {
    externalCorrelationId: <CorrelationId>,
    parentContainerId: <ParentContainerId>
}
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);
player.renderApp();
```

initPlayerByLogicalName

This method instantiates Player by LogicalName, EnvironmentId and TenantId and returns a player instance. This instance can be used to render an app or to subscribe to the Player events. It throws an error if valid options are not passed.

Syntax

```
public static initPlayerByLogicalName(playerOptions: PlayerOptions,
                                     logicalName: string,
```

```
environmentId: string,  
tenantId: string): Player;
```

Parameters

Name	Type	Required	Description
playerOptions	PlayerOptions	Yes	Describes the possible option that can be used to initialize a <i>Player</i> .
logicalName	String	Yes	Logical Name of the App. Apps that are part of solution would have the Logical name.
environmentId	String	Yes	Environment Id in which the app is created
tenantId	String	Yes	Tenant Id in which the app is created

How to get LogicalName, EnvironmentId, TenantId

LogicalName

1. Go to powerapps.com. Click on the Solutions link in the left-hand side navigation
2. Open the solution in which the App was added or created. Copy the "Name"

Solutions > PlayerSDKSolution

Display name	Name
PlayerSDKTestApp	crc1d_playersdktestapp_ec608

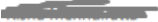
Environment Id

1. Go to powerapps.com. Copy the id after environments, highlighted in red.

<https://make.powerapps.com/environments/839eace6-59ab-4243-97ec-a5b8fcc104e4/apps/4e110cb7-afc4-416e-9dfe-3d0dd7d4400d/details>

Tenant Id

1. In powerapps.com, on the Apps tab, click or tap the ellipsis (...), then Details- copy the Tenant ID highlighted in red

Details	Versions	Connections	Flows	Analytics (preview)
Owner				
Created	8/18/2019, 10:18:00 PM			
Modified	8/19/2019, 1:24:11 PM			
Web link	https://apps.preview.powerapps.com/play/4e110cb7-afc4-416e-9dfe-3d0dd7d4400d?tenantId=72f988bf-86f1-41af-91ab-2d7cd011db47			
App ID	4e110cb7-afc4-416e-9dfe-3d0dd7d4400d			

Return Value

Returns a [Player](#) instance. If there are any failures while creating a new instance, an error is thrown.

Examples

Play a PowerApp by Logical Name

```
const options = {
    externalCorrelationId: <CorrelationId>,
    parentContainerId: <ParentContainerId>
}
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,
                                                         <environmentId>, <tenantId>);
player.renderApp();
```

renderApp

This method is used to play/render the PowerApp

Syntax

```
public renderApp():void;
```

Parameters

This method takes no parameters.

Return Value

It's a void function.

Examples

Play a PowerApp by App Id

```
const options = {
    externalCorrelationId: <CorrelationId>,
    parentContainerId: <ParentContainerId>
}
var player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);
player.renderApp();
```

Play a PowerApp by Logical Name

```
const options = {
    externalCorrelationId: <CorrelationId>,
    parentContainerId: <ParentContainerId>
}
var player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,
                                                         <environmentId>, <tenantId>);

player.renderApp();
```

dispose

This method disposes the player instance that was being used to play the App. It also, unsubscribes all the registered events.

Syntax

```
public dispose():void;
```

Parameters

This method takes no parameters.

Return Value

This method returns a void.

Examples

```
player.dispose();
```

setData

This method allows the host to set data on the host object that was created during the authoring time. This can be called once the app is loaded.

Syntax

```
public setData(data: PowerAppsData): void;
```

Parameters

Name	Type	Required	Description
data	PowerAppsData	Yes	Describes the possible option that can be used to set data on the App.

Return Value

It's a void function.

Examples

Set data when the app is loaded

```
// The schema for setting the data is set at authoring time using PowerAppsAuthoringSDK
const data: PowerAppsData = [
{
  Name: "XYZ"
},
{
  Name: "ABC",
}];

// this method should be called after the app is loaded.
// Subscribe to the onAppLoaded event and call it when that event is fired
player.setData(data);
```

registerHostCallback

This method allows the host to register methods that can be called in the App. Schema for the methods and data can be set using PowerApps Authoring SDK while creating the App.

Syntax

```
public registerHostFunction(name: string, func: Function) :void;
```

Parameters

Name	Type	Required	Description
name	string	Yes	Name of the function that's being registered
func	Function	Yes	Function that should be called in the app

Return Value

It's a void function.

Examples

Register host functions

```
// Method that needs to be registered
private GetRandomString() {
  return "XYZ";
}
```

```
player.registerHostFunction("GetRandomStringHostCallback", GetRandomString());
player.renderApp();
```

onAppLoaded

Event is triggered when App is successfully loaded. The *onAppLoaded* event is of type *EventHook< PlayerInfo>*. Which means when the event is raised, it has a single parameter which conforms to the *PlayerInfo* interface which provides the *appId*.

Syntax

```
public onAppLoaded: EventHook<PlayerInfo> = new EventHook(this);
```

Example

Subscribe/Unsubscribe to onAppLoaded event

```
// Use this to create a player instance by App Id
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);
// Use this to create a player instance by Logical Name
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,
                                                         <environmentId>, <tenantId>);

function onAppLoaded(playerInfo) {
    Console.log("User loaded an application with id: " + playerInfo.appId);
}

// Subscribe to app load event
player.onAppLoaded.subscribe(onAppLoaded);

// When the caller no longer wishes to receive callbacks unsubscribe from the event
player.onAppLoaded.unsubscribe(onAppLoaded);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
player.onAppLoaded.unsubscribeAll();
```

onAppUnload

Event is triggered when the App is closed. This can happen before load is fired. For e.g. It happens when user does not agree to consent dialog. The *onAppUnload* event is of type *EventHook< PlayerInfo>*.

Syntax

```
public onAppUnload: EventHook<PlayerInfo> = new EventHook(this);
```

Example

Subscribe/Unsubscribe to onAppUnload event

```
// Use this to create a player instance by App Id
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);
// Use this to create a player instance by Logical Name
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,
                                                         <environmentId>, <tenantId>);
```



```
function onAppUnload(playerInfo) {
    Console.log("App with id: " + playerInfo.appId + " Unloaded");
}

// Subscribe to app unload event
player.onAppUnload.subscribe(onAppUnload);

// When the caller no longer wishes to receive callbacks unsubscribe from the event
player.onAppUnload.unsubscribe(onAppUnload);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
player.onAppUnload.unsubscribeAll();
```

onAppError

Event is triggered when the App when does not load due to an error. The *onAppError* event is of type *EventHook<PlayerInfo>*.

Syntax

```
public onAppError: EventHook<PlayerInfo> = new EventHook(this);
```

Example

Subscribe/Unsubscribe to onAppError event

```
// Use this to create a player instance by App Id
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);
// Use this to create a player instance by Logical Name
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,
                                                         <environmentId>, <tenantId>);

function onAppError(playerInfo) {
    Console.log("App with id: " + playerInfo.appId + " errored");
}

// Subscribe to app error event
player.onAppError.subscribe(onAppError);

// When the caller no longer wishes to receive callbacks unsubscribe from the event
player.onAppError.unsubscribe(onAppUnload);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
player.onAppError.unsubscribeAll();
```

EventHook

The class implements the publisher or subscriber pattern so that listeners can subscribe to events using the subscribe method and unsubscribe using the unsubscribe method.

Methods

Method	Description
subscribe	Registers a function that conforms to the <i>EventListener</i> type that will be called when the event is raised.
unsubscribe	Registers a function that conforms to the <i>EventListener</i> type that will be called when the event is raised.
unsubscribeAll	If there are one or more listeners subscribed for an event, using <i>unsubscribeAll</i> will unsubscribe all the listeners.

subscribe / unsubscribe / unsubscribeAll

Syntax

```
eventHook.subscribe(listener);  
eventHook.unsubscribe(listener);  
eventHook.unsubscribeAll();
```

Parameters

Name	Type	Required	Description
Listener	EventListener<T>	Yes	A function that will be called when the event is raised.

Return Value

None

Example

Subscribe/Unsubscribe from Player Events

```
// Use this to create a player instance by App Id  
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <appId>);  
// Use this to create a player instance by Logical Name  
let player = PowerAppsPlayerSdk.Player.initPlayerByAppId(options, <logicalName>,  
                                                         <environmentId>, <tenantId>);  
  
function onAppUnload(playerInfo) {  
    Console.log("App with id: " + playerInfo.appId + " Unloaded");  
}  
  
// Subscribe to app unload event
```

```
player.onAppUnload.subscribe(onAppUnload);

// When the caller no longer wishes to receive callbacks unsubscribe from the event
player.onAppUnload.unsubscribe(onAppUnload);

// If there are multiple listeners, you can unsubscribe from all of them using
// unsubscribeAll method.
player.onAppUnload.unsubscribeAll();
```

Interfaces

SdkInitializer

This interface defines the contract for initializing the SDK, which must be done before using any other functionality from the SDK.

Syntax

```
export interface SdkInitializer {
    hostName: string;
}
```

Properties

Name	Type	Required	Description
hostname	string	Yes	This parameter is used primarily to help identify your host application when responding to support tickets.

RGBA

Describes the RGB color with optional value for alpha. This defines a type for splash screen color.

Syntax

```
export interface RGBA {
    r: number;
    g: number;
    b: number;
    a?: number;
}
```

Properties

Name	Type	Required	Description
r	number	Yes	Value for red color. Range 0-255
g	number	Yes	Value for green color. Range 0-255

b	number	Yes	Value for blue color. Range 0-255
a	number	No	Value for alpha. Range 1-100

PlayerOptions

Describes the possible options that can be used to play an App.

Syntax

```
export interface PlayerOptions {
    parentContainerId: string;
    externalCorrelationId: string;
    locale?: string;
    width?: string;
    height?: string;
    splashScreenColor?: RGBA;
    hideAppDetailsOnSplashScreen?: boolean;
    getAccessToken?: AccessTokenProvider;
}
```

Properties

Name	Type	Required	Description
parentContainerId	string	Yes	Id of the HTML in which the App should be rendered
externalCorrelationId	string	Yes	Id provided by the host app which could be used to correlate the events end to end
locale	string	No	Parameter to set the locale of the App
width	string	No	Sets the width of the Iframe. If not provided sets the width to 100%
height	string	No	Sets the height of the Iframe. If not provided sets the height to 100%
splashScreenColor	RGBA	No	Sets the color of the splash screen. If not provided default color would be set
hideAppDetailsOnSplashScreen	boolean	No	Hide showing the app splash screen (this hides the name of the app and icon). The loading dots will still be shown

getAccessToken	AccessTokenProvider	No	Hosts implementation that takes in an audience and calls either the tokenCallback with the access token for that audience, or the errorCallback if the access token cannot be retrieved. If this is not passed in, PowerApps will handle the authentication and may display pop ups.
----------------	-------------------------------------	----	--

PlayerInfo

This interface defines the set of properties that are available on objects returned by events (e.g., onAppLoaded, onAppUnload, onAppError) of the App.

Properties

Property	Description
appId	The application id of the application being played.

Enums

SdkState

This Enum describes the different states of SDK.

Name	Description
Uninitialized	Represents if the SDK is uninitialized
Initializing	Represents if the SDK is being initialized
Initialized	Represents if the SDK is initialized

Types

AccessTokenProvider

A type that takes in an audience, a tokenCallback with the access token for that audience, an errorCallback if the access token cannot be retrieved

Definition

```
export type AccessTokenProvider = (
  audience: string,
  tokenCallback: (token: string, parentRequestId?: string) => void,
  errorCallback: (errorMessage: string, parentRequestId?: string) => void,
  clientRequestId: string,
) => void;
```

RecordData

A type represents a record. This can be used to set data on a PowerApp.

Definition

```
export type RecordData = { [key: string]: string | number };
```

TableData

A type represents a table. This can be used to set data on a PowerApp.

Definition

```
export type TableData = RecordData[];
```

PowerAppsData

A generic PowerApp data object that can be used by the host app to set data on the App.

Definition

```
export type PowerAppsData = RecordData | TableData;
```

Existing Docs

PowerApps

- [PowerApps Overview](#)
- [Embed apps](#)
- [Custom connectors](#)
- [Download, Launch, and Param functions](#)

Power BI

- [Embedding](#)
- [Template apps](#)

Flow

[Embed Flow with apps](#)